Fine-tuning vs. In-context Learning in Large Language Models: A Formal Language Learning Perspective

Bishwamittra Ghosh[1], Soumi Das[1], Till Speicher[1], Qinyuan Wu[1], Mohammad Aflah Khan[1], Deepak Garg[1], Krishna P. Gummadi[1], Evimaria Terzi[2]
[1]Max Planck Institute for Software Systems, Germany, [2]Boston University, USA

Abstract

Large language models (LLMs) operate in two fundamental learning modes – *fine-tuning* (FT) and *in-context learning* (ICL) – raising key questions about which mode yields greater language proficiency and whether they differ in their inductive biases. Prior studies comparing FT and ICL have yielded mixed and inconclusive results due to inconsistent experimental setups. To enable a rigorous comparison, we propose a *formal language learning* task – offering precise language boundaries, controlled string sampling, and no data contamination – and introduce a *discriminative test* for language proficiency, where an LLM succeeds if it assigns higher generation probability to in-language strings than to out-of-language strings.

Empirically, we find that: (a) FT has greater language proficiency than ICL on in-distribution generalization, but both perform equally well on out-of-distribution generalization. (b) Their inductive biases, measured by the correlation in string generation probabilities, are similar when both modes partially learn the language but diverge at higher proficiency levels. (c) Unlike FT, ICL performance differs substantially across models of varying sizes and families and is sensitive to the token vocabulary of the language. Thus, our work demonstrates the promise of formal languages as a controlled testbed for evaluating LLMs, behaviors that are difficult to isolate in natural language datasets. Our source code is available at https://github.com/bishwamittra/formallm.

1 Introduction

Large language models (LLMs) operate in two fundamental learning modes: *fine-tuning* (FT) and *in-context learning* (ICL). FT simulates a closed-book exam, where LLMs learn by updating model parameters (Kaplan et al., 2020). ICL simulates an open-book exam, where LLMs learn from in-context examples without any parameter update (Brown et al., 2020). Both modes are widely applied in real-world tasks, including text summarization (Radford et al., 2019), question-answering (Yang et al., 2018), and conversational agents (Ouyang et al., 2022). A natural question is, therefore, which learning mode is *more language-proficient*, i.e., which mode recognizes patterns in the language better, and whether their *inductive bias*, i.e., the implicit assumptions about recognizing patterns, is similar or different. Despite its importance, this question remains open due to inconsistent experimental setups in prior studies.

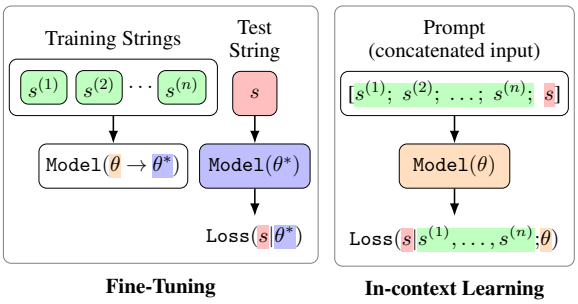


Figure 1: Fine-tuning and in-context learning are two learning modes of an LLM. In formal language learning, the learning task is to generate unseen strings from the language through syntactic pattern recognition (desideratum **D1**). Under an equal setting (desideratum **D2**), fine-tuning updates model parameters ($\theta \to \theta^*$) based on the training strings and computes the generation loss on a test string. In-context learning, however, takes a concatenated input prompt, where training strings serve as the prefix for generating the test string. Since the two learning modes differ in both input prompt and model parameters, a comparable evaluation metric is needed (desideratum **D3**).

Addressing this gap requires a principled experimental design.

Our Contributions. Our key contribution is the introduction of the following three-fold desiderata for comparing FT and ICL, and a controlled experimental framework that realizes these desiderata

Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 41651–41697
July 2-7, 2026

(see motivation in Figure 1). Several prior studies attempted to compare FT and ICL without satisfying all three desiderata, resulting in mixed and inconclusive results. Specifically, the closest to our work is Mosbach et al. (2023), who partially satisfy desiderata **D1** and **D2**, but fail to satisfy **D3**.

D1. Specification of the Learning Task: Syntax-focused Learning with Zero-prompting. We compare FT and ICL on learning a *probabilistic formal language*, which is a distribution of strings accepted by a probabilistic formal grammar (Manning, 2003; Chater and Manning, 2006). The learning task is to generate new strings based on recognizing syntactic patterns of the underlying language represented by the training strings (Section 3).

Formal languages offer several advantages for this comparison: (a) they contain syntax only, isolating syntactic pattern recognition from the semantic ambiguity of natural languages – the main focus of prior studies (Mosbach et al., 2023); (b) they provide full control over data distribution, enabling precise sampling of training and test strings and differentiation between in-distribution and out-of-distribution languages; (c) they are synthetic, avoiding data contamination and ensuring no model benefits from prior knowledge of the training data (Xu et al., 2024). These properties are difficult to guarantee with natural language datasets.

A practical challenge in comparing learning modes is *communicating the task via prompt-instructions*, which different LLMs may interpret differently (Wu et al., 2025; Zhao et al., 2021; Razavi et al., 2025; Zhuo et al., 2024). Formal language learning sidesteps this subjectivity: we consider a *zero-prompting* setup where the LLM only sees training strings and must generate new strings without any instruction.

D2. Allocation of Equal Resources. A fair comparison requires allocating equal resources to both FT and ICL[1]. We provide the same training and test data to both learning modes. Since FT and ICL have disjoint hyperparameters – batch-size, learning rates, and epochs for FT; example repetitions and inference temperature for ICL – we compare their best performance over respective hyperparameter settings, going beyond prior work (Mosbach et al., 2023; Yin et al., 2024) that addresses this only partially.

D3. Comparable Evaluation Metric. How can we evaluate the language proficiency of a learner in a language? There are two potential tests for language proficiency: generative and discriminative – the latter introduced in this work. The generative test computes the generation probability of in-language strings, but this is not directly comparable: LLMs vary in their priors, and FT and ICL of the same LLM but with different parameters treat the input prompt differently (Figure 1), making a direct numerical comparison of generation probability infeasible. The discriminative test instead checks whether in-language strings are generated with higher probability than *close yet grammatically incorrect* out-of-language strings, and results in a classification score – a metric that avoids model-specific and prompt-specific biases, making it comparable across both modes. Therefore, we claim that *the discriminative test is the appropriate metric for comparing* FT *and* ICL (Section 4).

Experimental Results. We experiment with 18 open-source LLMs from 6 model families and multiple formal languages, and reach the following conclusions: (a) Different LLMs converge to optimal FT performance, while their ICL ability varies substantially in formal languages. Model size contributes to improved performance in ICL but not in FT. (b) On in-distribution generalization, where training and test languages are the same, FT dominates ICL except in some LLMs where ICL is close to FT. On out-of-distribution generalization where training and test languages differ, both learning modes perform equally, and generalize only to out-of-distribution languages that are close to the training language. (c) The inductive bias, measured by the correlation of output generation probability of FT and ICL, is similar when both modes partially learn the language but diverges as proficiency improves with a higher number of examples. (d) FT is robust across languages, as assessed by varying the underlying grammar rules or token vocabulary. However, ICL performance is affected by the actual tokens used in the language.

Finally, we discuss the pitfalls of testing LLMs with natural language datasets, including imprecise sampling of training and test strings, data contamination, and ill-defined notions of in-distribution vs. out-of-distribution tasks, in Appendix D. Instead, we propose that synthetic formal languages are nec-

[1]Resource fairness includes data and compute fairness. The paper focuses on data fairness. Achieving compute fairness, however, is challenging, since FT and ICL incur costs at different stages: FT is costly during training, while ICL is costly at inference (Figure 41 in the Appendix).

essary for rigorous scientific study of LLMs, and that our work will inspire future research.

2 Motivation and Related Work

Here, we review related work and motivate why a comprehensive study comparing FT and ICL requires satisfying three desiderata: a precise specification of the learning task (**D1**), equal resource allocation (**D2**), and a comparable evaluation metric (**D3**). Prior work on comparing FT and ICL has largely overlooked one or more of these desiderata, yielding mixed and inconclusive results.

Independent Studies on FT and ICL. Several works independently investigate FT (Kaplan et al., 2020; Zhang et al., 2024; Hu et al., 2024) and ICL in LLMs (Shen et al., 2023; Reddy, 2024; Pan et al., 2023; Chen et al., 2025), and relate learning performance to model size, training data, etc. However, these studies examine FT and ICL in isolation rather than in a controlled head-to-head comparison, making it difficult to draw conclusions about their relative language proficiency. Moreover, these studies rely on natural language benchmarks, where pre-training can disproportionately affect FT and ICL performance due to data contamination – a confound our synthetic setup avoids.

Benchmarks. Concerning desideratum **D1**, natural language datasets (Rajpurkar et al., 2016; Kwiatkowski et al., 2019) often provide high-level descriptions of learning tasks, where in-distribution and out-of-distribution tasks are *less precisely defined*. Even within in-distribution tasks, there is no formal guarantee of coherence between training and test examples – unlike in a formal language, where all examples belong to the same language. Also, public datasets may result in data contamination, providing an unfair advantage to some LLMs (Dominguez-Olmedo et al., 2025). For example, we find both issues on the MNLI dataset (Williams et al., 2018), as previously studied by Mosbach et al. (2023) on comparing FT and ICL. Our results contradict their findings: on out-of-distribution tasks, FT and ICL perform equally well on formal languages, but FT is better than ICL on the MNLI dataset (Appendix D). The contradiction highlights the need for a well-defined learning task (desideratum **D1**).

Comparison of FT and ICL. The comparison between FT and ICL yields mixed conclusions, often due to violating desideratum **D2** on the equality of resources. Several studies conclude that FT outperforms ICL (Brown et al., 2020; Mosbach et al., 2023; Liu et al., 2022; Lester et al., 2021; Bhatia et al., 2023; Asai et al., 2024). However, the conclusions are based on using different model sizes, an unequal number of examples, and high variance across runs. Other studies find ICL better than FT (Yin et al., 2024; Bertsch et al., 2024; Kaneko et al., 2025; Soudani et al., 2024; Awadalla et al., 2022), as they typically employ suboptimal FT (e.g., 1 epoch/repetition), giving ICL an advantage.

Evaluation Metrics. A further gap in prior work is the absence of a comparable evaluation metric (desideratum **D3**). Existing studies rely on generative metrics such as cross-entropy loss or accuracy (Kallini et al., 2024; Jumelet and Zuidema, 2023; Bhattamishra et al., 2020; Wang, 2021; Akyürek et al., 2024), which are not directly comparable across learning modes: in FT, the loss is computed over the updated parameters θ^*, whereas in ICL, the loss is conditioned on in-context examples alongside the frozen parameters θ. As a result, a lower generative loss in one mode does not imply greater language proficiency relative to the other. We address this gap by introducing a *discriminative test* that evaluates whether a model assigns higher generation probability (equivalently, lower loss) to in-language strings than to out-of-language strings – a criterion that is comparable across both learning modes and model families (Section 4).

Formal Languages in LLM Research. Owing to their greater controllability, formal languages have been widely used to investigate the linguistic capabilities of LLMs (Jumelet and Zuidema, 2023), including their inductive biases in language learning (Papadimitriou and Jurafsky, 2023; White and Cotterell, 2021; Hopkins, 2022). Leveraging formal languages as a testbed, prior studies have compared the representational capacity of LLMs with various sequence-based models (Shi et al., 2022; Chi et al., 2023; Bhattamishra et al., 2020; Merrill, 2023; Strobl et al., 2023; Hahn, 2020), and analyzed the classes of formal languages that LLMs can learn (Delétang et al., 2023; Hahn and Rofin, 2024; Cotterell et al., 2018; Mielke et al., 2019; Borenstein et al., 2024). Notably, LLMs have been shown to learn hierarchical and probabilistic formal languages that mirror the recursive structure of natural language (Allen-Zhu and Li, 2023; Murty et al., 2023; Liu et al., 2023).

To our knowledge, no prior work has employed

$S \to A19$ [1]
$A19 \to A18\ A16$ [0.50]
$A19 \to A16\ A18\ A17$ [0.50]
$A18 \to A15\ A14\ A13$ [0.50]
$A18 \to A14\ A15\ A13$ [0.50]
$A17 \to A14\ A13\ A15$ [0.50]
$A17 \to A13\ A14\ A15$ [0.50]
$A16 \to A14\ A15$ [0.50]
$A16 \to A15\ A14$ [0.50]
$A15 \to A11\ A12\ A10$ [0.50]
$A15 \to A12\ A10\ A11$ [0.50]
$A14 \to A11\ A10\ A12$ [0.50]
$A14 \to A10\ A11\ A12$ [0.50]
$A13 \to A10\ A12\ A11$ [0.50]
$A13 \to A12\ A11\ A10$ [0.50]
$A12 \to 9\ 8\ 7$ [0.50]
$A12 \to 8\ 7$ [0.50]
$A11 \to 6\ 5$ [0.50]
$A11 \to 6\ 4\ 5$ [0.50]
$A10 \to 3\ 1$ [0.50]
$A10 \to 1\ 2\ 3$ [0.50]

Figure 2: Inspired by Allen-Zhu and Li (2023), we illustrate a hierarchical and probabilistic context-free grammar, representing language L_1. Here, non-terminals are marked in red, terminal tokens (alphabet) are in teal, and rule-probabilities are in blue. The grammar contains non-terminal symbols S and A's, alphabet $\mathbf{T} = \{1, 2, \ldots, 9\}$, and probabilistic production rules which are applied in a hierarchical way. To generate a string, we start from the non-terminal S and recursively apply production rules until reaching tokens in $\mathbf{T}$ only.

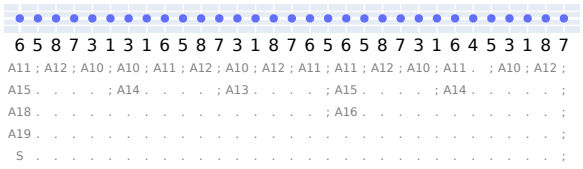


Figure 3: A string s generated by the grammar in Figure 2. The rule '$A19 \to A18\ A16$ [1]' indicates that non-terminal $A19$ is expanded to $A18$ followed by $A16$ with probability 1, and so on, until reaching $\mathbf{T}$. The generation probability of s is the multiplication of the probabilities of rules applied recursively to generate s, and $P(s) = (0.5)^{23}$.

formal languages to compare the language proficiency and inductive biases of learning modes of LLMs – this forms our central focus. Extended related work is in Appendix A.

3 Experimental Framework

We discuss preliminaries on formal languages, how to teach them to LLMs via different learning modes, and the experimental setup – all of which realize our desiderata.

Formal Languages. We use *probabilistic formal languages*, particularly the class generated by hierarchical probabilistic context free grammars (HPCFGs), as the learning task for LLMs (desideratum **D1**). HPCFGs capture the recursive structure of natural language. Formally, a probabilistic formal language L is defined on a set of tokens (alphabet) $\mathbf{T}$, and specifies a probability distribution P_L over strings, $P_L : \mathbf{T}^* \to [0, 1]$, where $\mathbf{T}^*$ is the set of all strings. A string s is *in-language* w.r.t. L if $P_L(s) > 0$, and *out-of-language* if $P_L(s) = 0$. $\mathbf{T}$ is a proper subset of the vocabulary $\mathbf{V}$ of all tokens of the LLM.

Languages. We consider six languages, denoted by $\{L_i\}_{i=1}^{6}$, based on a combination of two distinct HPCFGs and three distinct alphabet sets. For each language, we sample non-overlapping training ($n_{\text{train}} \in \{1, 2, 4, \ldots, 1024\}$) and test strings ($n_{\text{test}} = 1024$), following the distribution in a given language (desideratum **D2**). Figures 2 and 3 illustrate a representative grammar and a sampled string, respectively. Additional details on formal languages, respective grammars, the sampling process, and length distributions of generated strings are in Appendix B.

Construction of Out-of-language Strings. We quantify the *degree of incorrectness* of an out-of-language string as a *distance* from the language under investigation, which we use in the discriminative test in Section 4 (desideratum **D3**). We generate grammatically incorrect strings in two ways: (a) *Incorrect by edit*: We edit in-language strings to create out-of-language strings (through the addition, deletion, and replacement of tokens at random positions), where edit distance is the number of edits made to the in-language string. (b) *Incorrect by randomization*: We sample random strings over the language's alphabet, matching the length distribution of the language. On average, such random strings have a very high edit distance from the language. In both cases, we ensure non-membership of out-of-language strings via a grammar parser.

Teaching the Language to an LLM. To teach a language L to an LLM M, we sample strings from L and provide them to M via both learning modes. `FT` is performed for a fixed number of epochs, denoted by $m = 50$, where in each epoch the LLM iterates over the strings while minimizing cross-entropy loss. Formally, consider a dataset of n strings $D \triangleq \{s^{(j)}\}_{j=1}^{n}$ sampled from the language, $D \sim L$. For a given string s and its token s_i at the i-th position, let $P_M(s_i | s_{[1,i-1]})$ be the probability that the LLM M assigns to the token s_i given the prefix tokens $s_{[1,i-1]}$. The cross-entropy loss of the LLM M on the dataset D is the per-token negative log probability at every token position of all strings in D, $\texttt{loss}_M(D) \triangleq -\frac{1}{n} \sum_{s \in D} \frac{1}{|s|} \sum_{i=1}^{|s|} \log P_M(s_i \mid s_{[1,i-1]})$.

In ICL, we provide the same strings in D as in-context examples. Specifically, ICL takes a set of ordered examples $\langle s^{(1)}, \ldots, s^{(n)} \rangle$ as a prefix for a test string s. The ICL examples are concatenated using separators, such as semicolons, leading to a prompt $s^{(1)}[\texttt{sep}] \ldots [\texttt{sep}] s^{(n)}[\texttt{sep}] s$. Similar to epochs in FT, we repeat the examples in ICL a fixed number of times, $m \in \{1, 2, 4, 8, 16\}$. In both modes, we compare the language proficiency at the *optimal epoch or repetition* m^*, following desideratum **D2**.

Models. We study 18 open-source LLMs from 6 model families: Mistral (Jiang et al., 2023), Llama (Touvron et al., 2023a,b; Dubey et al., 2024), Qwen (Yang et al., 2024), Gemma (Mesnard et al., 2024; Riviere et al., 2024), Pythia (Biderman et al., 2023), and Opt (Zhang et al., 2022), ranging from 0.5B to 13B parameters. Each experiment is repeated three times by randomly sampling training strings with different seeds. Additional details on hyperparameters are provided in Appendix B.

4 The Test for Language Proficiency

Today, most prior language proficiency tests for LLMs are based on generative measures – how well an LLM generates strings belonging to the language. These tests, however, do not consider grammatically incorrect strings outside the language. Often, error patterns reveal more about language proficiency – two non-native speakers may have similar generative performance, but the types of mistakes they make reveal their underlying language prior. Our discriminative test is motivated by this analogy; moreover, it is comparable across learning modes, enabling a direct comparison between FT and ICL (desideratum **D3**).

The Generative Test. The generative test evaluates how well an LLM generates unseen test strings from the language – the higher the generation performance, the better the language proficiency.

Formally, consider two LLMs M and M' and a target language L. M and M' can also be two learning modes of the same LLM. Using the generative test, M is more language proficient in L than M', if M generates strings in L with a lower loss than M', i.e., $\texttt{loss}_M(L) < \texttt{loss}_{M'}(L)$.

Issues with the Generative Test. Two reasons hinder a direct comparison between FT and ICL using the generative test. (i) Absolute loss (perplexity or probability) is incomparable across LLMs:

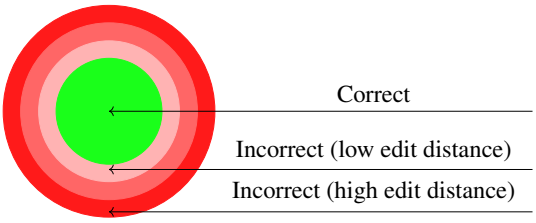
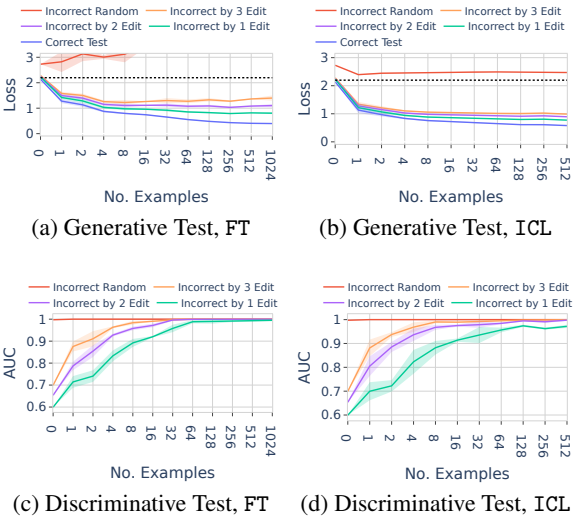


Figure 4: We visualize the set of all strings in a hierarchy, where the inner green circle denotes grammatically correct in-language strings, and the outer red circle denotes grammatically incorrect out-of-language strings. The generative test focuses on generation performance within the green circle, while the discriminative test focuses on comparative generation performance between green and red (especially at low edit distance) circles.

generation loss is impacted by pre-training setup, vocabulary, model parameters, random initialization, etc. As a result, different LLMs optimally trained on the same language may generate strings with different losses. (ii) FT and ICL result in different input prompts and require comparing the same LLM with different parameters (Figure 1). These confounding factors make direct comparison infeasible: if FT and ICL generate a string with different loss, we cannot determine whether the difference is due to different input prompts, model parameters, or both. To overcome these issues, we propose a discriminative test, which considers strings outside the language.

The Discriminative Test. The key intuition behind the discriminative test is that *if an LLM learned a language, it should generate strings in the language with lower loss than strings outside the language*. Thus, the discriminative test attempts to classify in-language and out-of-language strings based on their generation loss, where the success of classification implies language proficiency. As shown in Figure 4, the test can be made stricter by picking out-of-language strings *close* to in-language strings (according to some distance metric such as edit distance) and checking if they can still be identified as out-of-language.

Formally, let $\mathsf{T}(L)$ denote out-of-language strings, constructed by editing strings in L to ensure they are not in L. Consider a binary (linear) classifier, where the input is the generation loss assigned by an LLM to strings in $L \cup \mathsf{T}(L)$, and the classification task is to determine their membership. Let $\texttt{auc}_M(L, \mathsf{T}(L)) \in [0, 1]$ be the AUC (area under the receiver operating characteristic curve) of the classifier using model M; the higher the



(a) Generative Test, FT (b) Generative Test, ICL

(c) Discriminative Test, FT (d) Discriminative Test, ICL

Figure 5: Language proficiency of Mistral-7B on language L_1, while varying the number of examples in both learning modes.

value, the better. Thus, LLM M is more language proficient in L than M', if $\texttt{auc}_M(L, \mathsf{T}(L)) > \texttt{auc}_{M'}(L, \mathsf{T}(L))$. We formalize the comparability of the discriminative test in the following claim.

Claim 1. *For a given language, the discriminative test yields a numerically comparable score between two learning modes of an LLM and across LLMs, unlike the generative test.*

To support our claim, the discriminative test asks the same LLM or learning mode (i.e., equal parameters) to generate in-language and out-of-language strings, where all strings use the same input prompt format. Thus, the derived classification score is comparable across learning modes and LLMs (details in Appendix F).

Demonstration of Language Proficiency Tests. We now illustrate the behavior of both tests empirically. Figure 5 shows the language proficiency of an LLM w.r.t. the generative test (loss) in the top row and the discriminative test (AUC) in the bottom row, for both FT and ICL.

Observation 1. Generative test alone is misleading. In Figures 5a and 5b, with increasing examples, the loss decreases on in-language test strings, shown in the blue line. From this observation alone, we cannot determine whether language proficiency is achieved. Because, loss also decreases on out-of-language strings that are close, and there is often a loss-overlap between in-language and out-of-language strings, especially when the number of examples is low. Therefore, *the generative test alone is insufficient in determining whether language proficiency is achieved in the target language or in nearby languages.*

Observation 2. Discriminative test score is correlated with training size and edit distance of out-of-language strings. In Figures 5c and 5d, the AUC of the discriminator increases with the number of examples. Hence, the LLM becomes increasingly proficient in the language, by not only generating strings from the language with lower loss, but also distinguishing them from strings outside the language. Moreover, AUC is correlated with the edit distance of out-of-language strings; the higher the edit distance, the higher the AUC. *Importantly, the AUC scores of* FT *and* ICL *are comparable when both modes use the same number of examples and degree of grammatical incorrectness.*

In the next section, we apply the discriminative test to compare FT and ICL, and report the AUC for discriminating in-language test strings from out-of-language strings at edit distance 1 – the most stringent setting of the discriminative test.

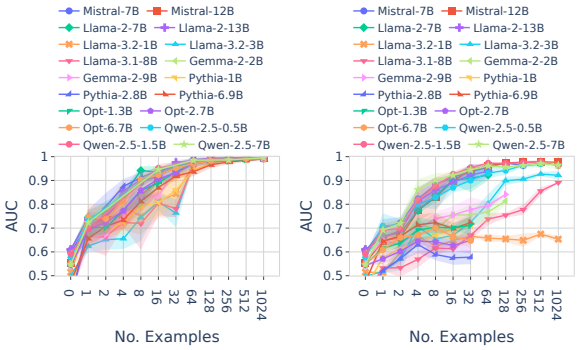
5 Fine-tuning vs. In-context Learning

We study the language proficiency of FT and ICL in LLMs by learning syntactic patterns from formal languages. Specifically, we address the following research questions.[2]

RQ1. When evaluating FT and ICL independently on a given language, how language proficient are different LLMs of varying sizes and families?
RQ2. Which learning mode is more language proficient when evaluated jointly on in-distribution and out-of-distribution generalization?
RQ3. Do FT and ICL result in similar inductive bias while learning a formal language?
RQ4. How robust is the performance of FT and ICL to changes in languages?

Answer to RQ1: Different LLMs attain a similar and near-optimal language proficiency under FT, but their ICL ability varies substantially. In Figure 6, we report the AUC of FT and ICL across LLMs and example sizes while learning language L_1. In both modes, AUC increases with examples, indicating better learning.

[2]Additional results including evaluation on natural language datasets, capability of utilizing full ICL context by different LLMs, and the detailed implications of research questions are in the Appendix.

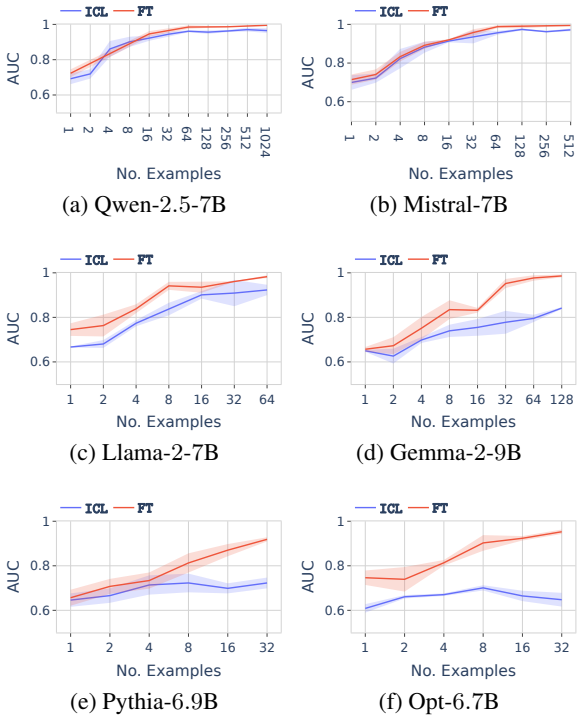


(a) FT (b) ICL

Figure 6: FT and ICL across different LLMs while learning language L_1. Different LLMs demonstrate similar FT performance, but their ICL ability varies.

ICL ability (AUC range)	Model
Good (≥ 0.75)	Qwen-2.5-7B, Mistral-7B, Qwen-2.5-1.5B, Llama-2-13B, Qwen-2.5-0.5B, Llama-2-7B, Mistral-12B
Moderate (≥ 0.6)	Gemma-2-2B, Gemma-2-9B, Pythia-6.9B, Opt-1.3B, Opt-6.7B, Pythia-1B, Llama-3.2-3B, Opt-2.7B, Llama-3.2-1B
Poor (< 0.6)	Llama-3.1-8B, Pythia-2.8B

Table 1: ICL ability of LLMs on language L_1 with up to 32 examples, based on discriminative AUC. In each group, LLMs are sorted in descending ICL ability.



(a) Qwen-2.5-7B (b) Mistral-7B

(c) Llama-2-7B (d) Gemma-2-9B

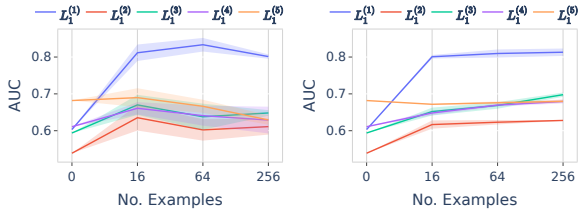
(e) Pythia-6.9B (f) Opt-6.7B

Figure 7: In-distribution generalization of FT vs. ICL on L_1 in comparable $\approx$ 7B parameter size LLMs. FT usually dominates ICL, except in Qwen-2.5-7B, Mistral-7B, and Llama-2-7B, where ICL is close to FT.

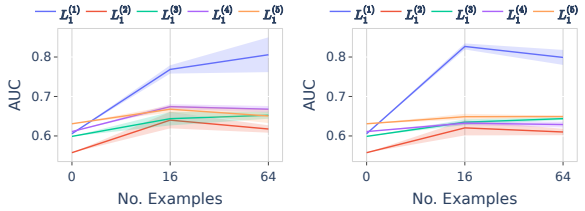
Fine-tuning. During FT, all models across families and parameter sizes eventually converge to the optimal AUC (> 0.99) after sufficient training examples, such as 512. Across example sizes $\{1, 16, 64, 256, 1024\}$, the average AUC of FT is similar across models: Llama-2 (0.93) > Qwen (0.92) > Mistral (0.91) > Opt (0.91) > Gemma (0.90) $\geq$ Pythia (0.90) > Llama-3 (0.88), where the respective AUC is inside the parentheses. Only in a few families (e.g., Opt) does the largest model achieve the highest AUC. In addition, a more proficient family often achieves its best language proficiency in an earlier epoch. For example, the median epoch is 7.5 for Llama-2, 12 for Opt, and 37 for Llama-3. *Therefore, different LLMs, regardless of sizes and families, can achieve similar language proficiency under* FT *on a tailored task like formal language learning.*

In-context Learning. In ICL, the AUC varies substantially within a model family and across model families. First, we observe that different LLMs have variable context length, restricting each model to a different maximum number of ICL examples. To compare all models fairly, we limit our analysis to 32 ICL examples, which all models can fit in their context. We find the following order of ICL ability of LLM families: Qwen (0.78) $\geq$ Mistral (0.78) > Llama-2 (0.77) > Gemma (0.69) > Opt (0.64) > Pythia (0.61) > Llama-3 (0.59). Due to variable performance, we propose a ranking of ICL ability of LLMs in Table 1. Importantly, within a family, ICL ability does not always correlate with model size (Mistral 7B > Mistral-12B) or model generations (Llama-2-7B > Llama-3.1-8B). Only in some families, such as Qwen, Pythia, and Llama-2, is the largest model better in ICL. Unlike FT, repeating ICL examples more than once worsens ICL performance: repeating examples takes up context space, and it is thus better to sample examples from the language distribution without repetition. *To conclude,* ICL *ability is more variable across LLMs, compared to* FT.

Answer to RQ2: On in-distribution language generalization, FT **dominates** ICL **in most LLMs; only in a subset of LLMs,** ICL **is close to** FT**. On out-of-distribution generalization, both** FT **and** ICL **perform similarly, and generalize well to the nearest language only.** In Figure 7, we compare FT and ICL of an LLM on in-distribution language generalization, where evaluation is performed on

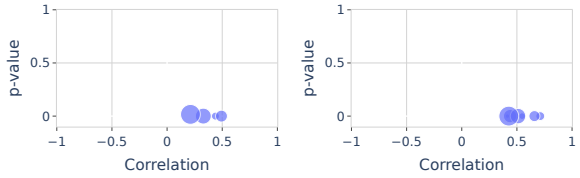


(a) FT, Mistral-7B (b) ICL, Mistral-7B

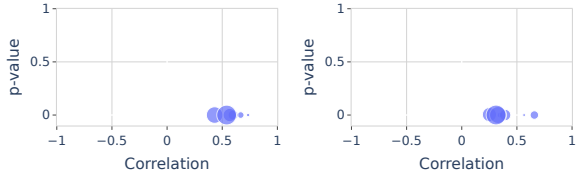


(c) FT, Llama-2-7B (d) ICL, Llama-2-7B

Figure 8: Out-of-distribution generalization of FT and ICL on increasingly distant languages, where both modes perform almost equally. L_1 is the base learned language, and generalization is performed on $L_1^{(\ell)}$, by changing ℓ rules in the grammar of L_1. $L_1^{(\ell)}$ contains all changed rules in $L_1^{(\ell-1)}$. Therefore, $\texttt{dist}(L_1, L_1^{(\ell-1)}) \leq \texttt{dist}(L_1, L_1^{(\ell)})$, where $2 \leq \ell \leq 5$ (see Eq. (1)).



(a) Qwen-2.5-7B (b) Mistral-7B



(c) Llama-2-7B (d) Gemma-2-9B

Figure 9: Inductive bias of ICL and FT, computed as the Pearson correlation of generation loss of FT and ICL on identical test strings. Correlation, despite being positive, tends to decrease with more examples (larger markers).

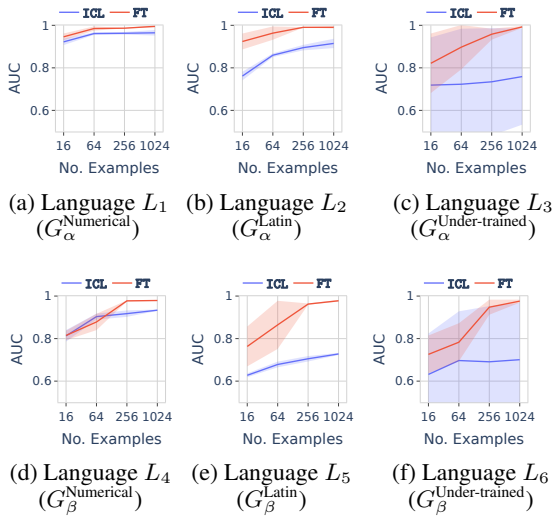
the same teaching language. In most LLMs, FT dominates ICL, and the performance difference becomes more pronounced with more examples. However, in a subset of models, such as Mistral-7B, Qwen-2.5-7B, and Llama-2-7B, ICL is close to FT – these models are usually ranked as having good ICL ability in Table 1. *Therefore,* FT *is more language proficient than* ICL *on in-distribution language generalization.*

For the comparison of FT and ICL on out-of-distribution generalization, the LLM first learns the language L_1, and then we evaluate the LLM on five other languages $\{L_1^{(1)}, \ldots, L_1^{(5)}\}$ of increasing distances from L_1 (Figure 8). We emphasize that formal languages offer a systematic distance computation between two languages (i.e., out-of-distribution tasks), unlike natural language datasets (Appendix D). Surprisingly, both modes perform similarly on out-of-distribution languages, and only perform well on the nearest language $L_1^{(1)}$. *Therefore, the superiority of* FT *over* ICL *on in-distribution generalization does not extend to out-of-distribution generalization.*

Answer to RQ3: The inductive bias of FT and ICL is often similar, but not equal. Similarity decreases with training examples. To compare the inductive bias of FT and ICL, we do not focus on how each mode operates internally, but on the correlation between their generation losses when evaluated on the same set of strings. Thus, if correlation is high, inductive bias is similar, since both modes find the language similarly easy or difficult to generate. In Figure 9, the Pearson correlation is positive (< 0.8). However, correlation tends to decrease with more training examples, implying that as each mode learns the language better, they do so differently. *To summarize, the inductive bias of* FT *and* ICL *is often similar, but similarity decreases as each mode learns the language better with more training examples.*

Answer to RQ4: FT is more robust to changes in languages than ICL. In Figure 10, we study the robustness of FT and ICL on different languages, by changing the underlying grammar rules: G_α and G_β, and the alphabet: numerical, Latin, and under-trained tokens. FT is better than ICL in all languages, consistent with results in in-distribution generalization (Figure 7). Importantly, changing only the tokens (across columns) introduces more variability than changing the grammar rules (across rows), and the variability is more pronounced in ICL than FT. For example, when considering under-trained tokens, i.e., tokens barely seen in pre-training (Land and Bartolo, 2024), ICL performance is the worst. *Therefore, for robust performance,* FT *is preferred over* ICL.

We further validate our findings **beyond formal languages to natural languages**, as studied



(a) Language L_1 ($G_\alpha^{\text{Numerical}}$) (b) Language L_2 (G_α^{Latin}) (c) Language L_3 ($G_\alpha^{\text{Under-trained}}$)

(d) Language L_4 ($G_\beta^{\text{Numerical}}$) (e) Language L_5 (G_β^{Latin}) (f) Language L_6 ($G_\beta^{\text{Under-trained}}$)

Figure 10: Robustness of language proficiency of FT and ICL in Qwen-2.5-7B while varying languages in two ways: changing the grammar rules (rows) and changing the alphabet tokens (columns). The underlying grammar for a language is inside the parentheses. Compared to FT, ICL is sensitive to the tokens used in the language, despite having the same underlying grammar.

by Mosbach et al. (2023) (Appendix D). These findings hold for natural language on in-distribution generalization, but not for out-of-distribution generalization. In doing so, we identify several issues in natural language datasets such as data contamination and a poor differentiation between in-distribution and out-of-distribution tasks, factors that we carefully avoid in formal languages.

Key Implications of the Study

We draw the following implications from our study (details in Appendix G):

- FT is better than ICL if the test and training languages are the same. ICL is, however, preferred on out-of-distribution languages, where general language understanding of the model is retained as parameters are not updated.
- FT and ICL are likely to recognize patterns similarly when few examples are given (i.e., before the language is well learned). With more examples, the inductive bias of FT and ICL usually differs.
- Within a family, a larger model size can lead to better ICL, but not necessarily better FT. Among model families, Qwen, Mistral, and Llama-2 are better in both modes.
- Unlike FT, ICL is more token-sensitive, despite following the same grammar rules. Since models in the same family may have different pretraining recipes impacting the same tokens differently, we may expect variability in ICL within a family (Mistral-7B $>$ Mistral-12B, Llama-2-7B $>$ Llama-3.1-8B). However, if the target language contains less-common (under-trained) tokens, FT is preferred.
- The discriminative test considers strings outside the language in determining language proficiency. If LLM M assigns higher generation loss to strings in L than LLM M', but the discriminative AUC of M is higher than that of M', we still expect M to be more proficient in L, contrary to what generation loss alone would suggest. Here, M may have numerically high generation loss possibly due to model-specific priors, but its higher ability to differentiate strings inside and outside the language makes it more proficient.

6 Conclusion

We study the language proficiency and inductive bias of FT and ICL – two fundamental learning modes in LLMs. We propose three desiderata for a fair comparison between learning modes, which prior studies overlooked. To satisfy these desiderata, we consider the task of formal language learning by an LLM, and propose a comparable discriminative test for evaluating language proficiency.

Our controlled experimental framework leads to important findings: FT is better than ICL on in-distribution language generalization, but both perform equally on out-of-distribution generalization. Their inductive bias is similar, but this similarity decreases as both modes learn the language better with more training examples. Unlike FT, ICL performance is more sensitive to the tokens used in the language, even with the same grammar rules.

Many of our results on synthetic formal languages are difficult to achieve with poorly controlled natural language datasets. More broadly, formal language learning opens up the possibility of evaluating LLMs in a controlled testbed, enabling precise study of their capabilities beyond what natural language datasets afford.

Limitations

Despite the precise controllability of our formal language setup and the utility of our discriminative test as a comparative metric between FT and

ICL, this work has limitations that warrant further investigation.

Formal languages are limited to context-free languages. The paper focuses on hierarchical context-free languages, which mimic the recursive structure of natural languages. However, we highlight the need for further study to confirm our findings in other classes of formal languages, such as regular and context-sensitive languages.

Scope of LLMs. Our goal is to compare FT and ICL on LLMs of equal parameter size. Since FT is more compute-intensive, we limit our experiments to a maximum of 13B parameter size models. Moreover, we do not perform an extensive hyperparameter search in FT, such as batch size and learning rate. Rather, we find the optimal epoch for each FT run and compare it with the optimal repetition of examples in ICL. Furthermore, we restrict experiments to full fine-tuning, while acknowledging that several parameter-efficient fine-tuning methods exist and may lead to different conclusions. We experiment with the non-instruction-tuned models, since a formal language learning task with only syntactic pattern recognition does not require instructions – comparing FT and ICL on instruction-tuned models is left for future work.

Larger models ($>$ 13B) may have better in-context learning performance. Does it invalidate our results? Since ICL is inferior to FT on in-distribution performance, a natural question is whether considering larger models would further improve ICL. While we expect ICL to improve with model-size, so does FT, and our finding that FT is better than ICL on in-distribution languages remains unchanged.

We find variable ICL performance across LLMs. How can we explain this? To explain the variability of ICL performance, we have conducted two studies: (a) determining whether existing LLMs utilize their full ICL context (see Appendix E), and (b) identifying the sensitivity of ICL to tokens used in our experiments (see **RQ4** in Section 5). The former result identifies which LLMs fully utilize their ICL context and which do not. The latter result shows that the tokens used for experimentation have a large impact on ICL performance, and the same set of tokens may have been pre-trained to varying degrees across LLMs. While these results are important, we leave a more informed explanation of model-specific ICL performance for future work.

Inductive bias comparison is based only on the generative test. We measure inductive bias via generation loss on individual strings. Extending this to a discriminative test requires per-string discrimination: we define a string as *learned* if the LLM assigns it lower loss than all its out-of-language neighbors – making it a local minimum. Inductive bias then reduces to the correlation of per-string discrimination, which we leave for future work.

Ethics Statement

The paper investigates how different learning modes of large language models (LLMs), namely fine-tuning (FT) and in-context learning (ICL), compare in their language proficiency and inductive bias. Our experiments involve controlled and synthetically generated formal languages with no human subject involvement or use of private data. As such, the research study does not present immediate ethical risks from the data collection or model training processes. Our scientific results have profound implications for choosing the right mode of learning for LLMs in various applications.

References

Ekin Akyürek, Bailin Wang, Yoon Kim, and Jacob Andreas. 2024. In-context language learning: Architectures and algorithms. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org.

Zeyuan Allen-Zhu and Yuanzhi Li. 2023. Physics of language models: Part 1, learning hierarchical language structures. *arXiv preprint arXiv:2305.13673*.

Akari Asai, Sneha Kudugunta, Xinyan Yu, Terra Blevins, Hila Gonen, Machel Reid, Yulia Tsvetkov, Sebastian Ruder, and Hannaneh Hajishirzi. 2024. BUFFET: Benchmarking large language models for few-shot cross-lingual transfer. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 1771–1800, Mexico City, Mexico. Association for Computational Linguistics.

Anas Awadalla, Mitchell Wortsman, Gabriel Ilharco, Sewon Min, Ian Magnusson, Hannaneh Hajishirzi, and Ludwig Schmidt. 2022. Exploring the landscape of distributional robustness for question answering models. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Amanda Bertsch, Maor Ivgi, Uri Alon, Jonathan Berant, Matthew R. Gormley, and Graham Neubig. 2024. In-context learning with long-context models: An

in-depth exploration. In *First Workshop on Long-Context Foundation Models @ ICML 2024*.

Kush Bhatia, Avanika Narayan, Christopher M De Sa, and Christopher Ré. 2023. TART: A plug-and-play transformer module for task-agnostic reasoning. *Advances in Neural Information Processing Systems*, 36:9751–9788.

Satwik Bhattamishra, Kabir Ahuja, and Navin Goyal. 2020. On the ability and limitations of transformers to recognize formal languages. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, Online. Association for Computational Linguistics.

Stella Biderman, Hailey Schoelkopf, Quentin Gregory Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, and 1 others. 2023. Pythia: A suite for analyzing large language models across training and scaling. In *International Conference on Machine Learning*, pages 2397–2430. PMLR.

Nadav Borenstein, Anej Svete, Robin Chan, Josef Valvoda, Franz Nowak, Isabelle Augenstein, Eleanor Chodroff, and Ryan Cotterell. 2024. What languages are easy to language-model? a perspective from learning probabilistic regular languages. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, and 1 others. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Nick Chater and Christopher D Manning. 2006. Probabilistic models of language processing and acquisition. *Trends in cognitive sciences*, 10(7):335–344.

Wentong Chen, Yankai Lin, ZhenHao Zhou, HongYun Huang, YanTao Jia, Zhao Cao, and Ji-Rong Wen. 2025. ICLEval: Evaluating in-context learning ability of large language models. In *Proceedings of the 31st International Conference on Computational Linguistics*, pages 10398–10422, Abu Dhabi, UAE. Association for Computational Linguistics.

Ta-Chung Chi, Ting-Han Fan, Alexander I Rudnicky, and Peter J Ramadge. 2023. Transformer working memory enables regular language reasoning and natural language length extrapolation. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, Singapore. Association for Computational Linguistics.

Noam Chomsky. 1956. Three models for the description of language. *IRE Transactions on information theory*, 2(3):113–124.

Michael Collins. 2013. Probabilistic context-free grammars (PCFGs).

Ryan Cotterell, Sabrina J Mielke, Jason Eisner, and Brian Roark. 2018. Are all languages equally hard to language-model? In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, New Orleans, Louisiana. Association for Computational Linguistics.

Colin de la Higuera, James Scicluna, and Mark-Jan Nederhof. 2014. On the computation of distances for probabilistic context-free grammars. *arXiv preprint arXiv:1407.1513*.

Grégoire Delétang, Anian Ruoss, Jordi Grau-Moya, Tim Genewein, Li Kevin Wenliang, Elliot Catt, Chris Cundy, Marcus Hutter, Shane Legg, Joel Veness, and 1 others. 2023. Neural networks and the chomsky hierarchy. In *The Eleventh International Conference on Learning Representations*.

Ricardo Dominguez-Olmedo, Florian E Dorner, and Moritz Hardt. 2025. Training on the test task confounds evaluation and emergence. In *The Thirteenth International Conference on Learning Representations*.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, and 1 others. 2024. The Llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Himanshu Gupta, Saurabh Arjun Sawant, Swaroop Mishra, Mutsumi Nakamura, Arindam Mitra, Santosh Mashetty, and Chitta Baral. 2023. Instruction tuned models are quick learners. *arXiv preprint arXiv:2306.05539*.

Michael Hahn. 2020. Theoretical limitations of self-attention in neural sequence models. *Transactions of the Association for Computational Linguistics*, 8:156–171.

Michael Hahn and Mark Rofin. 2024. Why are sensitive functions hard for transformers? In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Bangkok, Thailand. Association for Computational Linguistics.

Mark Hopkins. 2022. Towards more natural artificial languages. In *Proceedings of the 26th Conference on Computational Natural Language Learning (CoNLL)*, pages 85–94.

Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

Shengding Hu, Yuge Tu, Xu Han, Ganqu Cui, Chaoqun He, Weilin Zhao, Xiang Long, Zhi Zheng, Yewei Fang, Yuxiang Huang, Xinrong Zhang, Zhen Leng Thai, Chongyi Wang, Yuan Yao, Chenyang Zhao, Jie

Zhou, Jie Cai, Zhongwu Zhai, Ning Ding, and 5 others. 2024. MiniCPM: Unveiling the potential of small language models with scalable training strategies. In *First Conference on Language Modeling*.

Thomas F Icard. 2020. Calibrating generative models: The probabilistic Chomsky–Schützenberger hierarchy. *Journal of Mathematical Psychology*, 95:102308.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7b.

Jaap Jumelet and Willem Zuidema. 2023. Transparency at the source: Evaluating and interpreting language models with access to the true distribution. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, Singapore. Association for Computational Linguistics.

Julie Kallini, Isabel Papadimitriou, Richard Futrell, Kyle Mahowald, and Christopher Potts. 2024. Mission: Impossible language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Bangkok, Thailand. Association for Computational Linguistics.

Masahiro Kaneko, Danushka Bollegala, and Timothy Baldwin. 2025. The gaps between fine tuning and in-context learning in bias evaluation and debiasing. In *Proceedings of the 31st International Conference on Computational Linguistics*, pages 2758–2764.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Matthew Kelcey, Jacob Devlin, Kenton Lee, Kristina N. Toutanova, Llion Jones, Ming-Wei Chang, Andrew Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. 2019. Natural questions: a benchmark for question answering research. *Transactions of the Association of Computational Linguistics*.

Sander Land and Max Bartolo. 2024. Fishing for Magikarp: Automatically detecting under-trained tokens in large language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, Miami, Florida, USA. Association for Computational Linguistics.

Teven Le Scao and Alexander M Rush. 2021. How many data points is a prompt worth? In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2627–2636.

Eric Lehman, Evan Hernandez, Diwakar Mahajan, Jonas Wulff, Micah J Smith, Zachary Ziegler, Daniel Nadler, Peter Szolovits, Alistair Johnson, and Emily Alsentzer. 2023. Do we still need clinical language models? In *Conference on health, inference, and learning*, pages 578–597. PMLR.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Ziqian Lin and Kangwook Lee. 2024. Dual operating modes of in-context learning. In *ICLR 2024 Workshop on Mathematical and Empirical Understanding of Foundation Models*.

Bingbin Liu, Jordan T Ash, Surbhi Goel, Akshay Krishnamurthy, and Cyril Zhang. 2023. Transformers learn shortcuts to automata. In *The Eleventh International Conference on Learning Representations*.

Haokun Liu, Derek Tam, Mohammed Muqeeth, Jay Mohta, Tenghao Huang, Mohit Bansal, and Colin A Raffel. 2022. Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning. *Advances in Neural Information Processing Systems*, 35:1950–1965.

Christopher D Manning. 2003. Probabilistic syntax. *Probabilistic linguistics*, 289341.

William Merrill. 2023. Formal languages and the NLP black box. In *International Conference on Developments in Language Theory*, pages 1–8. Springer.

Thomas Mesnard, Cassidy Hardin, Robert Dadashi, Surya Bhupatiraju, Shreya Pathak, Laurent Sifre, Morgane Rivière, Mihir Sanjay Kale, Juliette Love, Pouya Tafti, Léonard Hussenot, Pier Giuseppe Sessa, Aakanksha Chowdhery, Adam Roberts, Aditya Barua, Alex Botev, Alex Castro-Ros, Ambrose Slone, Amélie Héliou, and 88 others. 2024. Gemma: Open models based on Gemini research and technology. *arXiv preprint arXiv:2403.08295*.

Sabrina J Mielke, Ryan Cotterell, Kyle Gorman, Brian Roark, and Jason Eisner. 2019. What kind of language is hard to language-model? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, Florence, Italy. Association for Computational Linguistics.

Marius Mosbach, Tiago Pimentel, Shauli Ravfogel, Dietrich Klakow, and Yanai Elazar. 2023. Few-shot fine-tuning vs. in-context learning: A fair comparison and evaluation. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 12284–12314, Toronto, Canada. Association for Computational Linguistics.

Shikhar Murty, Pratyusha Sharma, Jacob Andreas, and Christopher D Manning. 2023. Characterizing intrinsic compositionality in transformers with tree projections. In *The Eleventh International Conference on Learning Representations*.

Michael Oliver and Guan Wang. 2024. Crafting efficient fine-tuning strategies for large language models. *arXiv preprint arXiv:2407.13906*.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, and 1 others. 2022. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744.

Jane Pan, Tianyu Gao, Howard Chen, and Danqi Chen. 2023. What in-context learning "learns" in-context: Disentangling task recognition and task learning. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 8298–8319, Toronto, Canada. Association for Computational Linguistics.

Isabel Papadimitriou and Dan Jurafsky. 2023. Injecting structural hints: Using language models to study inductive biases in language learning. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, Singapore. Association for Computational Linguistics.

Branislav Pecher, Ivan Srba, and Maria Bielikova. 2025. Comparing specialised small and general large language models on text classification: 100 labelled samples to achieve break-even performance. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 165–184.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas. Association for Computational Linguistics.

Shauli Ravfogel, Yoav Goldberg, and Tal Linzen. 2019. Studying the inductive biases of RNNs with synthetic variations of natural languages. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, Minneapolis, Minnesota. Association for Computational Linguistics.

Amirhossein Razavi, Mina Soltangheis, Negar Arabzadeh, Sara Salamat, Morteza Zihayat, and Ebrahim Bagheri. 2025. Benchmarking prompt sensitivity in large language models. In *European Conference on Information Retrieval*, pages 303–313. Springer.

Gautam Reddy. 2024. The mechanistic basis of data dependence and abrupt learning in an in-context classification task. In *The Twelfth International Conference on Learning Representations*.

Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, Léonard Hussenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ramé, Johan Ferret, Peter Liu, Pouya Tafti, Abe Friesen, Michelle Casbon, Sabela Ramos, Ravin Kumar, Charline Le Lan, Sammy Jerome, Anton Tsitsulin, and 178 others. 2024. Gemma 2: Improving open language models at a practical size. *arXiv preprint arXiv:2408.00118*.

Lingfeng Shen, Aayush Mishra, and Daniel Khashabi. 2023. Do pretrained transformers really learn in-context by gradient descent? *arXiv preprint arXiv:2310.08540*.

Hui Shi, Sicun Gao, Yuandong Tian, Xinyun Chen, and Jishen Zhao. 2022. Learning bounded context-free-grammar via LSTM and the transformer: difference and the explanations. In *Proceedings of the AAAI conference on artificial intelligence*, volume 36, pages 8267–8276.

Heydar Soudani, Evangelos Kanoulas, and Faegheh Hasibi. 2024. Fine tuning vs. retrieval augmented generation for less popular knowledge. In *Proceedings of the 2024 Annual International ACM SIGIR Conference on Research and Development in Information Retrieval in the Asia Pacific Region*, pages 12–22.

Krishna Prasad Varadarajan Srinivasan, Prasanth Gumpena, Madhusudhana Yattapu, and Vishal H Brahmbhatt. 2024. Comparative analysis of different efficient fine tuning methods of large language models (LLMs) in low-resource setting. *arXiv preprint arXiv:2405.13181*.

Lena Strobl, William Merrill, Gail Weiss, David Chiang, and Dana Angluin. 2023. Transformers as recognizers of formal languages: A survey on expressivity. *arXiv preprint arXiv:2311.00208*.

Hongjin Su, Jungo Kasai, Chen Henry Wu, Weijia Shi, Tianlu Wang, Jiayi Xin, Rui Zhang, Mari Ostendorf, Luke Zettlemoyer, Noah A Smith, and 1 others. 2023. Selective annotation makes language models better few-shot learners. In *The Eleventh International Conference on Learning Representations*.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, and 1 others. 2023a. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, and 1 others. 2023b. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Shunjie Wang. 2021. *Evaluating transformer's ability to learn mildly context-sensitive languages*. University of Washington.

Jerry Wei, Jason Wei, Yi Tay, Dustin Tran, Albert Webson, Yifeng Lu, Xinyun Chen, Hanxiao Liu, Da Huang, Denny Zhou, and 1 others. 2023. Larger language models do in-context learning differently. *arXiv preprint arXiv:2303.03846*.

Jennifer C White and Ryan Cotterell. 2021. Examining the inductive bias of neural language models with artificial languages. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, Online. Association for Computational Linguistics.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122. Association for Computational Linguistics.

Qinyuan Wu, Mohammad Aflah Khan, Soumi Das, Vedant Nanda, Bishwamittra Ghosh, Camila Kolling, Till Speicher, Laurent Bindschaedler, Krishna Gummadi, and Evimaria Terzi. 2025. Towards reliable latent knowledge estimation in llms: Zero-prompt many-shot based factual knowledge extraction. In *Proceedings of the Eighteenth ACM International Conference on Web Search and Data Mining*, pages 754–763.

Cheng Xu, Shuhao Guan, Derek Greene, M Kechadi, and 1 others. 2024. Benchmark data contamination of large language models: A survey. *arXiv preprint arXiv:2406.04244*.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, and 1 others. 2024. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2369–2380, Brussels, Belgium. Association for Computational Linguistics.

Qingyu Yin, Xuzheng He, Chak Tou Leong, Fan Wang, Yanzhao Yan, Xiaoyu Shen, and Qiang Zhang. 2024. Deeper insights without updates: The power of in-context learning over fine-tuning. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 4138–4151, Miami, Florida, USA. Association for Computational Linguistics.

Biao Zhang, Zhongtao Liu, Colin Cherry, and Orhan Firat. 2024. When scaling meets LLM finetuning: The effect of data, model and finetuning method. In *The Twelfth International Conference on Learning Representations*.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, and 1 others. 2022. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*.

Zihao Zhao, Eric Wallace, Shi Feng, Dan Klein, and Sameer Singh. 2021. Calibrate before use: Improving few-shot performance of language models. In *International conference on machine learning*, pages 12697–12706. PMLR.

Jingming Zhuo, Songyang Zhang, Xinyu Fang, Haodong Duan, Dahua Lin, and Kai Chen. 2024. ProSA: Assessing and understanding the prompt sensitivity of llms. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 1950–1976.

A Extended Related Work

This section reviews two bodies of related work: studies on FT and ICL as learning modes, and prior use of formal languages in the context of LLMs.

A.1 Learning Modes in LLM: Fine-tuning and In-context Learning

We discuss existing studies that independently investigate fine-tuning and in-context learning, followed by their direct comparison.

Fine-tuning: A number of works including Kaplan et al. (2020); Zhang et al. (2024); Hu et al. (2024); Srinivasan et al. (2024); Oliver and Wang (2024); Hu et al. (2022) study the effects of fine-tuning or its variants with respect to model scaling, where larger fine-tuned models with less data outperform smaller models with the same amount of data, leading to compute-efficient training. Our experiments on synthetic formal languages do not demonstrate such a pattern, possibly because we allow all models of different sizes to reach their optimal fine-tuning performance on formal languages, where there is no tangible benefit of being large.

In-context learning: Given a set of examples as demonstrations, ICL allows LLMs to extract patterns without updating model parameters. Several studies attempt to explain how learning is achieved in ICL, by comparing it to gradient descent (Shen et al., 2023), in-weights learning (Reddy, 2024), and Boolean function learning in a controlled setting (Bhattamishra et al., 2020). Recently, Pan et al. (2023); Lin and Lee (2024) explore the dual characteristics of ICL: (i) task learning, where the test examples are unseen during pre-training, and (ii) task recognition/retrieval, where test examples are seen during the pre-training, and LLMs are asked to retrieve them using a different prompt.

A related question is how ICL performance scales with model size. Wei et al. (2023) study the relationship between ICL and model scale, where overriding semantic priors like flipping labels improves with larger models. In contrast, Chen et al. (2025) observe that ICL ability does not linearly correlate with model size. Our study finds that in the majority of model families, model size improves ICL, while in a few families, a medium-sized model performs better at ICL.

Fine-tuning versus In-context learning. Several works compare FT and ICL, with inconclusive results. Brown et al. (2020); Mosbach et al. (2023); Liu et al. (2022); Lester et al. (2021); Bhatia et al. (2023); Asai et al. (2024) agree that FT is better than ICL. However, this observation is made under unequal conditions, violating desideratum **D2**: using incomparable models (Liu et al., 2022), unequal numbers of examples (Brown et al., 2020; Liu et al., 2022), and observing high variance across different choices of examples (Asai et al., 2024).

Another group of works, including Yin et al. (2024); Bertsch et al. (2024); Kaneko et al. (2025); Soudani et al. (2024); Awadalla et al. (2022), finds that ICL is better than FT. To the best of our knowledge, none of these works fine-tune the models to their optimal point, e.g., Yin et al. (2024) fine-tune for only 1 epoch, and Awadalla et al. (2022) fine-tune for 10 epochs. The inconsistencies in experimental designs reinforce the need for desideratum **D2**, where different modes of learning are given a fair comparison under an equal allocation of resources.

Several works further compound this issue by comparing FT and ICL across models of different sizes, occasionally using different datasets. For example, Pecher et al. (2025); Lehman et al. (2023) investigate whether smaller FT models are better than larger general-purpose models adapted via ICL. Su et al. (2023) showcase the usefulness of selective annotation, but when comparing between FT and ICL, they use different model sizes: smaller models for FT vs larger models for ICL. Gupta et al. (2023) compare among ICL, instruction-tuning, and FT, where ICL and instruction-tuning are conducted on the same model, but FT is performed on a different larger model. Furthermore, the data used for instruction-tuning are not given to ICL. Finally, Le Scao and Rush (2021) compare FT and ICL on an identical masked language model – this model is fundamentally different from autoregressive LLMs in terms of generating tokens, which is our focus. In all the papers, there is still a chance of data contamination, which we have carefully avoided using formal language learning.

A.2 Formal Languages and LLMs

To address the data contamination and experimental inconsistencies inherent in natural language testbeds, we ground our comparison in formal languages. Many prior works have studied formal languages in the context of LLMs, focusing primarily on the expressive power of language models and the learnability of formal grammars — neither of which directly addresses our goal of comparing FT and ICL.

What is the relative representation capability of LLMs compared to other sequence models, or more specifically, what classes of languages are learnable by an LLM? LLMs with a transformer architecture may have a different representation capability than other neural language models, such as LSTMs and RNNs. We refer to a recent survey discussing the expressiveness of LLMs as a language recognizer (Strobl et al., 2023). Towards comparing representation capability, Shi et al. (2022) find that both LSTMs and transformer networks can simulate context-free languages with bounded recursion, suggesting similar representational power between the two. However, unlike transformers, LSTMs fail to decompose the latent representation space. (Bhattamishra et al., 2020) observe a clear contrast between the performance of transformers and LSTMs on regular languages. They find that in comparison with LSTMs, transformers achieve limited performance on languages involving periodicity, modular counting, and even simpler star-free variants of Dyck-1 languages. Delétang et al. (2023) explore how neural network models used for program induction relate to the idealized computational models defined by the Chomsky hierarchy (Chomsky, 1956). They find that neural language models are hard to place on the standard Chomsky hierarchy. Several works criticize their setup, since they consider a language transduction task (mapping one language to another), which is different from the language recognition task (Icard, 2020). Borenstein et al. (2024) consider learning strings from deterministic and probabilistic finite state automata. They empirically test the learnability as a function of various complexity parameters of the language and the hidden state size of the transformer and RNN. In a different line of work, (Akyürek et al., 2024) evaluate neural networks' abilities to learn regular languages in `ICL`. Rather than learning one particular distribution from the training dataset, they infer the generating mechanism using `ICL`. Similar to Delétang et al. (2023), they find that RNNs are better suited to modeling formal languages than transformers. Kallini et al. (2024) construct a continuum of languages that differ in their hardness to learn and show that GPT-2 has difficulty in learning the carefully constructed impossible languages, compared to English.

While most of the works in this line capture the expressiveness of LLMs and their differing representation ability with other sequence models, we fundamentally criticize their evaluation metrics. As elaborated in Section 4, they focus on testing how well an LLM learns the grammar rules or states of the automata, whereas our discriminative measure is rule/state agnostic and focuses on whether the LLMs can generate strings from the language better than strings outside the language – LLMs may learn in a way different from specific rules/states, and it is non-trivial to measure this.

Can LLMs learn the underlying structure of formal languages, and if so, how? Several studies utilize the controlled data generation of formal languages to study different NLP (natural language processing) aspects of the LLM. Formal languages, particularly those derived from context-free grammars, can imitate the rich recursive structure of natural languages. Therefore, many studies focus on teaching the LLM strings from a formal language and explaining how LLMs might learn them (Allen-Zhu and Li, 2023; Murty et al., 2023; Liu et al., 2023). In another line, Jumelet and Zuidema (2023) study if causal and masked LLMs capture the true underlying patterns if trained on a true distribution. They find that causal LLMs approximate the theoretically optimal perplexity of the PCFG more closely than masked LLMs. Along this direction, several studies consider the known distribution to analyze the impact of topological features of a language (Cotterell et al., 2018; Mielke et al., 2019; Ravfogel et al., 2019; Mielke et al., 2019; Papadimitriou and Jurafsky, 2023; White and Cotterell, 2021). Several studies propose augmenting LLMs with additional components to enable them to learn certain classes of languages with ease. For example, Chi et al. (2023) propose to add working memory, such as weight sharing, adaptive-Depth, and sliding-dilated attention to a GPT model to enable it to learn the parity function, which is hard for an LLM to learn (Hahn and Rofin, 2024). In contrast to this line of work, our focus is to apply formal languages to study different modes of learning in LLMs: `FT` and `ICL`, which, to the best of our knowledge, is novel.

B Extended Experimental Setup

All experiments are conducted in compute clusters with Python as the programming language (version 3.10), where we use 8x Nvidia H100 94GB NVL GPUs and 2x AMD EPYC 9554 CPU @ 3.1 GHz, 2x64 cores, and 24x96GB RAM. `FT` is performed with a batch size of 8 and a linear learning rate scheduler with a warm-up ratio of 0.05. We fix the

learning rate to 5×10^{-5} for the Qwen, Gemma, and Llama-3 families; 5×10^{-6} for the Mistral, Opt, and Llama-2 families; and 10^{-5} for the Pythia family. During inference, we sidestep temperature sampling and instead record the loss of each target token given the preceding tokens in the string.

Below, we provide details of the formal languages used in our experiments, along with their formal definitions. Intuitively, we carefully design languages to show the robustness of our results by changing the grammar rules and token types.

Formal Languages and Grammars. Throughout our experiments, we provide the LLM with strings sampled from a probabilistic formal language. Formally, a probabilistic formal language is represented by a *probabilistic formal grammar*, or simply *grammars* (Collins, 2013). A grammar consists of two sets of symbols called the *non-terminals* and *terminals*, a set of production rules for rewriting strings that contain at least one non-terminal, and a probability distribution over the production rules. More precisely, a probabilistic formal grammar is defined as a quintuple.

$$G \triangleq (\mathbf{N}, \mathbf{T}, \mathbf{R}, S, \mathbf{P})$$

where $\mathbf{N}$ is the set of non-terminals, $\mathbf{T}$ is the set of terminals (equivalently, tokens), $\mathbf{R}$ is the set of production rules, $S \in \mathbf{N}$ is the start non-terminal, and $\mathbf{P}$ is the set of probabilities on production rules.

Formal languages are divided into well-known classes based on the *complexity* of the language membership problem, i.e., the *complexity* of the grammars needed to generate them (Chomsky, 1956). In this paper, we use one class of grammars, namely, hierarchical probabilistic context-free grammars (HPCFGs) (Allen-Zhu and Li, 2023). Specifically, our experiments are based on teaching LLMs languages represented by HPCFGs, which are syntactically simple and can represent languages that are structurally similar to natural languages (Allen-Zhu and Li, 2023; Shi et al., 2022).

Description of Grammars and Identified Languages. In our experiments, we consider two generic structures for the considered grammars, one adapted from Allen-Zhu and Li (2023), namely G_α, and another proposed by us, namely G_β. We propose variants of these grammars by considering different alphabet sets.

In Figure 11, in the first generic structure G_α, each grammar has $\mathbf{N} = \{S, A10, A11, \ldots, A19\}$ and $\mathbf{T} = \{1, 2, 3, \ldots, 9\}$. The grammar has four levels of hierarchy: the non-terminals from top to bottom levels are $\{A19\}$, $\{A16, A17, A18\}$, $\{A13, A14, A15\}$, and $\{A10, A11, A12\}$, followed by terminals $\{1, 2, 3, \ldots, 9\}$. Since the terminals are derived from numerical characters, we call this grammar $G_\alpha^{\text{Numerical}}$; and if the terminals are derived from Latin characters, we call this grammar G_α^{Latin}. Each non-terminal (except the start non-terminal) has two expansion rules, consisting of non-terminals from the immediate lower level. Further, the expansion rules are probabilistic, where the sum of probabilities of all expansion rules from a given non-terminal is 1.

In Figure 12, the second generic structure G_β is inspired by bridging two HPCFGs together, and simulating long-range dependencies within the generated strings. Specifically, the sub-grammar rooted at $B4$ and the sub-grammar rooted at $E4$ are connected by non-terminal $C1_i$; and $E4$ ends with $T1_j$. Long-range dependencies are communicated through $C1_i$ and $T1_j$, by enforcing $i = j$ at each expansion of $S5$.

Table 2 shows the mapping of notations between grammars and identified languages. Figure 13 shows the length distribution of generated strings from different languages. Figure 14 demonstrates how hierarchical non-terminals are applied in different positions in the representative strings.

Sampling Strings from a Formal Language. Given a language L generated by an HPCFG, our objective is to sample a set of i.i.d. strings from the language. To *sample a string from the language*, we start from a special string in the grammar containing a single, distinguished nonterminal called the "start" or "root" symbol, and apply the production rules to rewrite the string repeatedly. If several rules can be used to rewrite the string at any stage, we sample one such rule from the probability distribution over the rules and apply it. We stop when we obtain a string containing terminals only. This string is a sample drawn from the language. We can repeat this process to draw any number of i.i.d. samples from the language.

In our experiments, we aim to split the sampled strings into disjoint training and test sets that have a similar probability distribution over string occurrences. To realize this goal, we first sample a finite number of strings from the language. We then perform a stratified split: we iterate over unique strings in random order, alternately assigning all

$S \rightarrow A19$ [1]
$A19 \rightarrow A18\ A16$ [0.50]
$A19 \rightarrow A16\ A18\ A17$ [0.50]
$A18 \rightarrow A15\ A14\ A13$ [0.50]
$A18 \rightarrow A14\ A15\ A13$ [0.50]
$A17 \rightarrow A14\ A13\ A15$ [0.50]
$A17 \rightarrow A13\ A14\ A15$ [0.50]
$A16 \rightarrow A14\ A15$ [0.50]
$A16 \rightarrow A15\ A14$ [0.50]
$A15 \rightarrow A11\ A12\ A10$ [0.50]
$A15 \rightarrow A12\ A10\ A11$ [0.50]
$A14 \rightarrow A11\ A10\ A12$ [0.50]
$A14 \rightarrow A10\ A11\ A12$ [0.50]
$A13 \rightarrow A10\ A12\ A11$ [0.50]
$A13 \rightarrow A12\ A11\ A10$ [0.50]
$A12 \rightarrow 9\ 8\ 7$ [0.50]
$A12 \rightarrow 8\ 7$ [0.50]
$A11 \rightarrow 6\ 5$ [0.50]
$A11 \rightarrow 6\ 4\ 5$ [0.50]
$A10 \rightarrow 3\ 1$ [0.50]
$A10 \rightarrow 1\ 2\ 3$ [0.50]

$S \rightarrow A19$ [1]
$A19 \rightarrow A18\ A16$ [0.50]
$A19 \rightarrow A16\ A18\ A17$ [0.50]
$A18 \rightarrow A15\ A14\ A13$ [0.50]
$A18 \rightarrow A14\ A15\ A13$ [0.50]
$A17 \rightarrow A14\ A13\ A15$ [0.50]
$A17 \rightarrow A13\ A14\ A15$ [0.50]
$A16 \rightarrow A14\ A15$ [0.50]
$A16 \rightarrow A15\ A14$ [0.50]
$A15 \rightarrow A11\ A12\ A10$ [0.50]
$A15 \rightarrow A12\ A10\ A11$ [0.50]
$A14 \rightarrow A11\ A10\ A12$ [0.50]
$A14 \rightarrow A10\ A11\ A12$ [0.50]
$A13 \rightarrow A10\ A12\ A11$ [0.50]
$A13 \rightarrow A12\ A11\ A10$ [0.50]
$A12 \rightarrow i\ h\ g$ [0.50]
$A12 \rightarrow h\ g$ [0.50]
$A11 \rightarrow f\ e$ [0.50]
$A11 \rightarrow f\ d\ e$ [0.50]
$A10 \rightarrow c\ a$ [0.50]
$A10 \rightarrow a\ b\ c$ [0.50]

Figure 11: Production rules of $G_{\alpha}^{\text{Numerical}}$ (left) and $G_{\alpha}^{\text{Latin}}$ (right).

occurrences of each string to the training or test set. This process repeats until the initial finite set is exhausted.

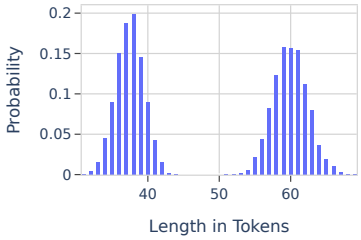
$S \rightarrow S5$ [1]
$S5 \rightarrow B4\ C1_1\ E4\ T1_1$ [0.25]
$S5 \rightarrow B4\ C1_2\ E4\ T1_2$ [0.25]
$S5 \rightarrow B4\ C1_3\ E4\ T1_3$ [0.25]
$S5 \rightarrow B4\ C1_4\ E4\ T1_4$ [0.25]
$B4 \rightarrow B3$ [0.3333]
$B4 \rightarrow B3\ B3\ B3$ [0.3333]
$B4 \rightarrow B3\ B3$ [0.3333]
$B3 \rightarrow B2$ [0.3333]
$B3 \rightarrow B2$ [0.3333]
$B3 \rightarrow B2\ B2$ [0.3333]
$B2 \rightarrow B1$ [0.3333]
$B2 \rightarrow B1$ [0.3333]
$B2 \rightarrow B1\ B1\ B1$ [0.3333]
$B1 \rightarrow 2\ 9\ 3$ [0.3333]
$B1 \rightarrow 9\ 6\ 1$ [0.3333]
$B1 \rightarrow 1\ 8\ 6\ 2$ [0.3333]
$E4 \rightarrow E3$ [0.3333]
$E4 \rightarrow E3\ E3$ [0.3333]
$E4 \rightarrow E3\ E3\ E3$ [0.3333]
$E3 \rightarrow E2$ [0.3333]
$E3 \rightarrow E2\ E2$ [0.3333]
$E3 \rightarrow E2$ [0.3333]
$E2 \rightarrow E1\ E1$ [0.3333]
$E2 \rightarrow E1$ [0.3333]
$E2 \rightarrow E1\ E1\ E1$ [0.3333]
$E1 \rightarrow 5\ 6$ [0.3333]
$E1 \rightarrow 1\ 8\ 6\ 6$ [0.3333]
$E1 \rightarrow 1\ 5\ 1\ 5\ 5\ 9$ [0.3333]
$T1_1 \rightarrow 1$ [1]
$T1_2 \rightarrow 2$ [1]
$T1_3 \rightarrow 3$ [1]
$T1_4 \rightarrow 4$ [1]
$C1_1 \rightarrow 5$ [1]
$C1_2 \rightarrow 6$ [1]
$C1_3 \rightarrow 7$ [1]
$C1_4 \rightarrow 8$ [1]
$C1_5 \rightarrow 9$ [1]

$S \rightarrow S5$ [1]
$S5 \rightarrow B4\ C1_1\ E4\ T1_1$ [0.25]
$S5 \rightarrow B4\ C1_2\ E4\ T1_2$ [0.25]
$S5 \rightarrow B4\ C1_3\ E4\ T1_3$ [0.25]
$S5 \rightarrow B4\ C1_4\ E4\ T1_4$ [0.25]
$B4 \rightarrow B3$ [0.3333]
$B4 \rightarrow B3\ B3\ B3$ [0.3333]
$B4 \rightarrow B3\ B3$ [0.3333]
$B3 \rightarrow B2$ [0.3333]
$B3 \rightarrow B2$ [0.3333]
$B3 \rightarrow B2\ B2$ [0.3333]
$B2 \rightarrow B1$ [0.3333]
$B2 \rightarrow B1$ [0.3333]
$B2 \rightarrow B1\ B1\ B1$ [0.3333]
$B1 \rightarrow b\ i\ c$ [0.3333]
$B1 \rightarrow i\ f\ a$ [0.3333]
$B1 \rightarrow a\ h\ f\ b$ [0.3333]
$E4 \rightarrow E3$ [0.3333]
$E4 \rightarrow E3\ E3$ [0.3333]
$E4 \rightarrow E3\ E3\ E3$ [0.3333]
$E3 \rightarrow E2$ [0.3333]
$E3 \rightarrow E2\ E2$ [0.3333]
$E3 \rightarrow E2$ [0.3333]
$E2 \rightarrow E1\ E1$ [0.3333]
$E2 \rightarrow E1$ [0.3333]
$E2 \rightarrow E1\ E1\ E1$ [0.3333]
$E1 \rightarrow e\ f$ [0.3333]
$E1 \rightarrow a\ h\ f\ f$ [0.3333]
$E1 \rightarrow a\ e\ a\ e\ e\ i$ [0.3333]
$T1_1 \rightarrow a$ [1]
$T1_2 \rightarrow b$ [1]
$T1_3 \rightarrow c$ [1]
$T1_4 \rightarrow d$ [1]
$C1_1 \rightarrow e$ [1]
$C1_2 \rightarrow f$ [1]
$C1_3 \rightarrow g$ [1]
$C1_4 \rightarrow h$ [1]
$C1_5 \rightarrow i$ [1]

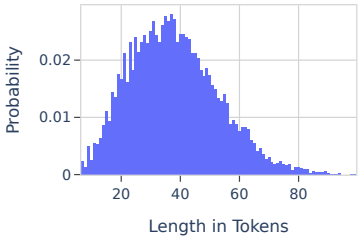
Figure 12: Production rules of $G_\beta^{\text{Numerical}}$ (left) and G_β^{Latin} (right).

Table 2: Notations of grammars and identified languages.

Grammar	Identified Language
$G_\alpha^{\text{Numerical}}$	L_1
G_α^{Latin}	L_2
$G_\alpha^{\text{Under-trained-tokens}}$	L_3
$G_\beta^{\text{Numerical}}$	L_4
G_β^{Latin}	L_5
$G_\beta^{\text{Under-trained-tokens}}$	L_6



(a) L_1 (also L_2, L_3)

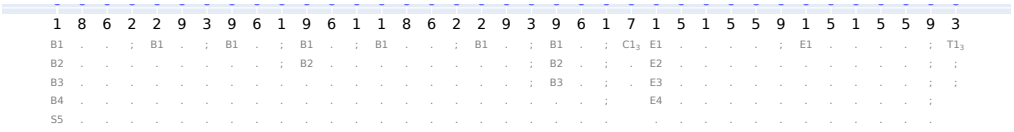


(b) L_4 (also L_5, L_6)

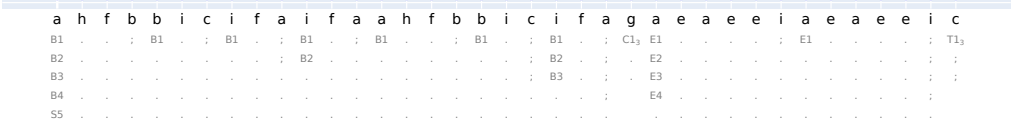
Figure 13: Length distribution of considered probabilistic languages, based on 10000 sampled strings per language.

(a) Language L_1 (Grammar $G_\alpha^{\text{Numerical}}$)

(b) Language L_2 (Grammar G_α^{Latin})



(c) Language L_4 (Grammar $G_\beta^{\text{Numerical}}$)



(d) Language L_5 (Grammar G_β^{Latin})

Figure 14: Representative strings from different languages, annotated with non-terminals applied in different positions by the respective hierarchical grammar.

Distance Between Languages. In probabilistic languages, a well-known approach to compute language distance is to compare the distribution of strings generated by both languages (de la Higuera et al., 2014). In our implementation, we choose a simplified distance metric based on L_2-norm.

$$\texttt{dist}_{L_2}(L_1, L_2) = \sqrt{\Sigma_{s \in \mathbf{T}^*}(P_{L_1}(s) - P_{L_2}(s))^2} \tag{1}$$

While distance metrics have their nuances, our goal is to systematically modify the original language, more specifically the underlying grammar, such that we can intuitively interpret language distance, irrespective of the distance metric used.

For simulating out-of-distribution generalization of learning modes, we modify the base grammar $G_\alpha^{\text{Numerical}}$ (abbreviated as G) in the following way: We construct five grammars $\{G^{(1)}, \ldots, G^{(5)}\}$ by perturbing $\ell \in \{1, 2, 3, 4, 5\}$ production rules of G, such that $G^{(\ell)}$ contains all perturbed production rules in $G^{(\ell-1)}$. The order in which rule-perturbation is applied is the following:

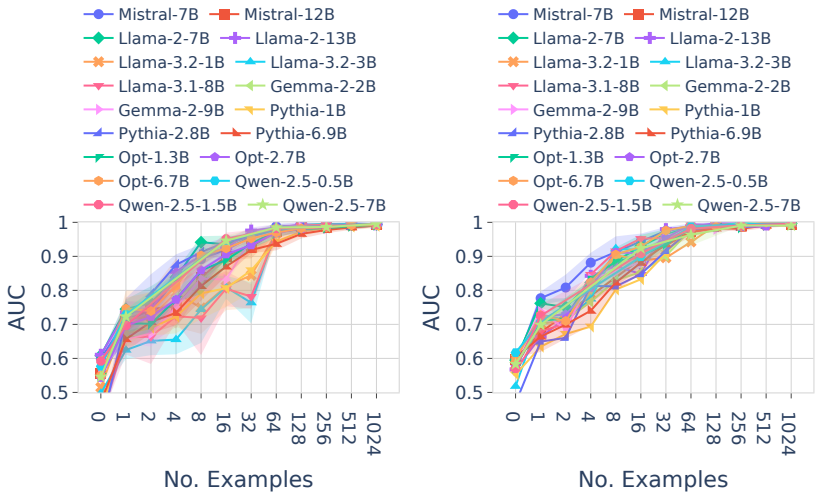
$$A10^{(1)} \rightarrow 1\ 3\ 2\ [0.50]$$
$$A11^{(2)} \rightarrow 5\ 6\ [0.50]$$
$$A12^{(3)} \rightarrow 8\ 7\ 9\ [0.50]$$
$$A10^{(4)} \rightarrow 3\ 1\ [0.50]$$
$$A12^{(5)} \rightarrow 8\ 7\ [0.50]$$

Intuitively, $G^{(1)}$ contains perturbed rule $\{A10^{(1)}\}$, $G^{(2)}$ contains perturbed rule $\{A10^{(1)}, A11^{(2)}\}$, and so on. Finally, each grammar $G^{(\ell)}$ identifies a language $L^{(\ell)}$ in Figure 8.

C Additional Experimental Results

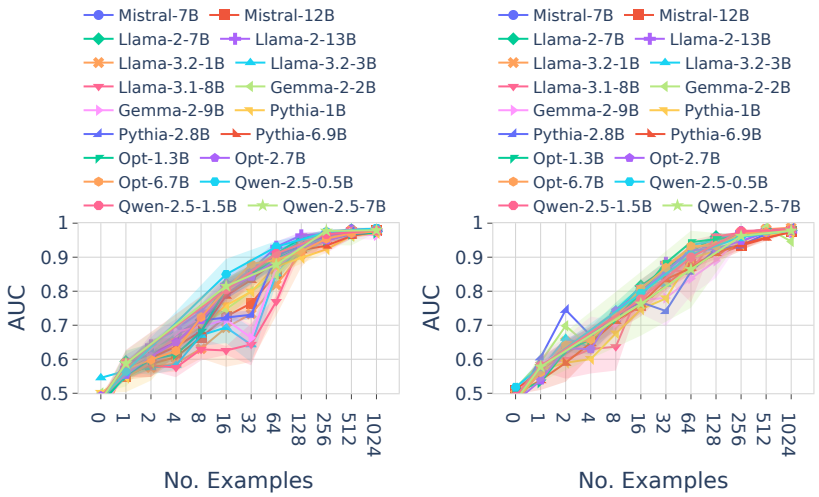
In the following, we outline additional experimental results.

- Independent evaluation of `FT` and `ICL` on different languages across datasets in Figure 15, 16.
- Intra-family `FT` and `ICL` performance in Figure 17, 18.
- Robustness of `FT` and `ICL` of individual models across languages in Figure 19, 20 21, 22.
- Inductive bias of LLMs across languages in Figure 23, 24, 25, 26.
- Out-of-distribution generalization on languages of different distances in Figure 27.
- `FT` vs. `ICL` on natural language datasets in Appendix D.
- Evaluating the utilization of the full context for `ICL` in Appendix E.
- Generative vs. discriminative tests for determining language proficiency in Appendix F.
- Comparison of different learning modes across compute cost in Figure 41.



(a) Language L_1

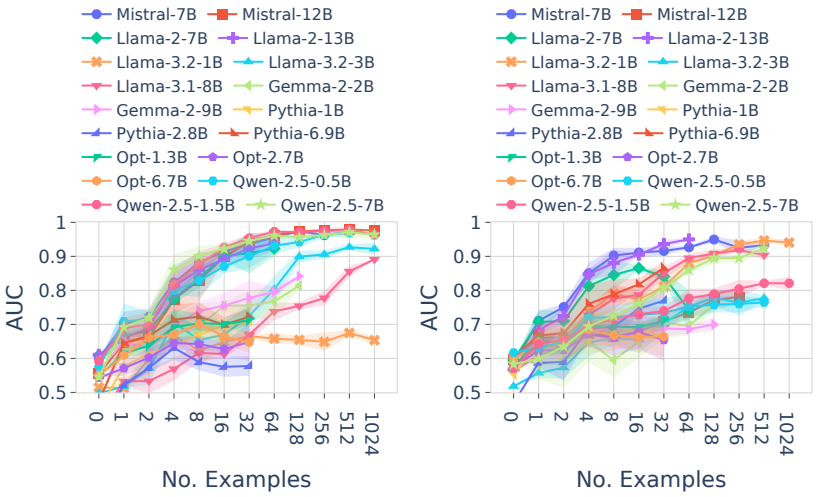
(b) Language L_2



(c) Language L_4

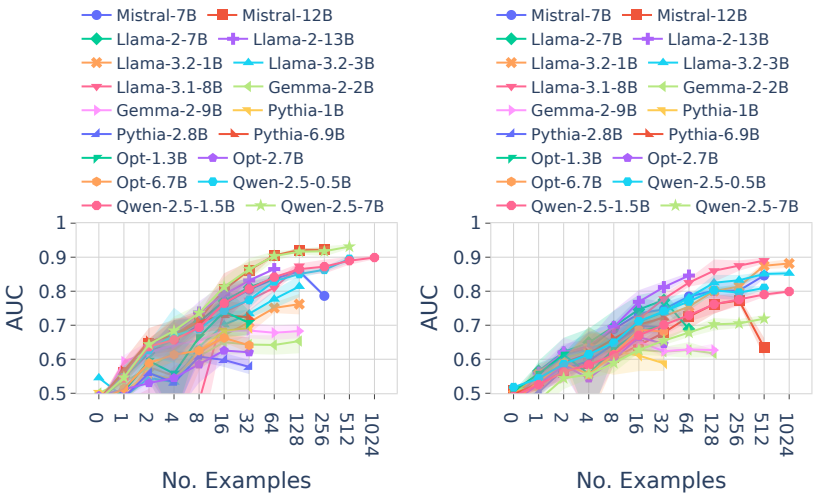
(d) Language L_5

Figure 15: Optimal fine-tuning performance in all models across different languages.



(a) Language L_1

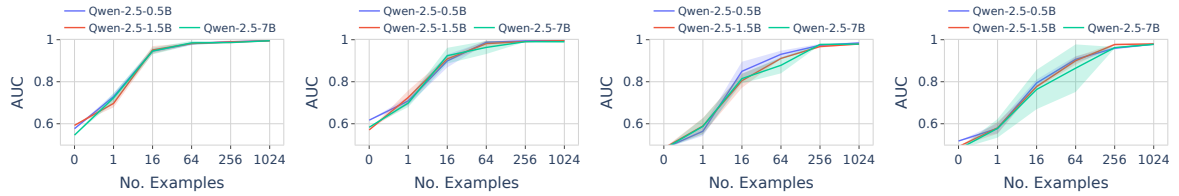
(b) Language L_2



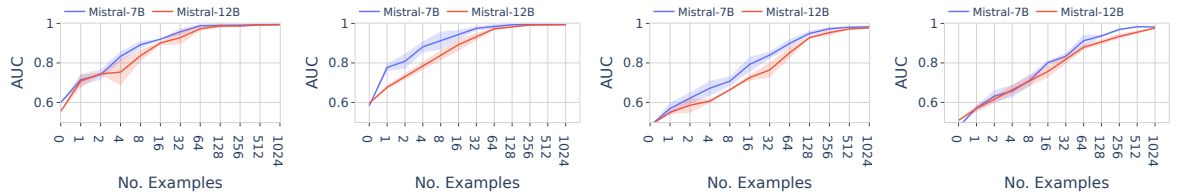
(c) Language L_4

(d) Language L_5

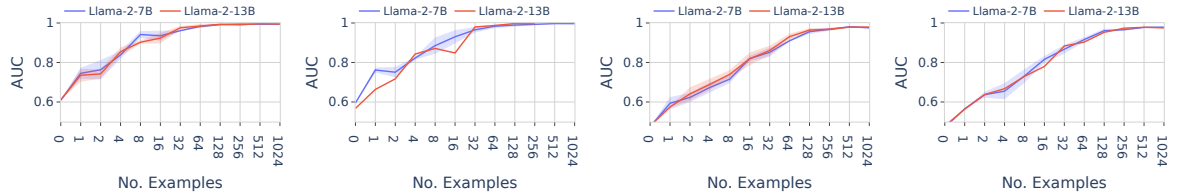
Figure 16: In-context learning performance of all models across different languages



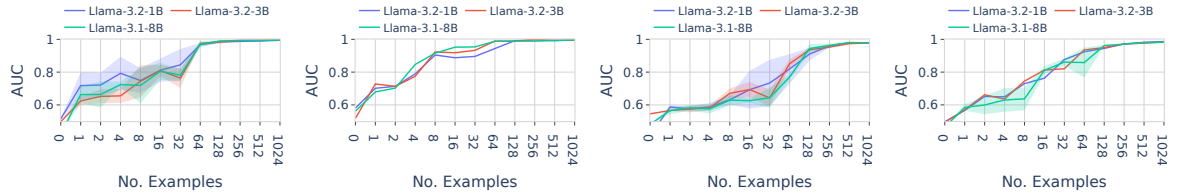
Qwen, Language L_1 | Qwen, Language L_2 | Qwen, Language L_4 | Qwen, Language L_5



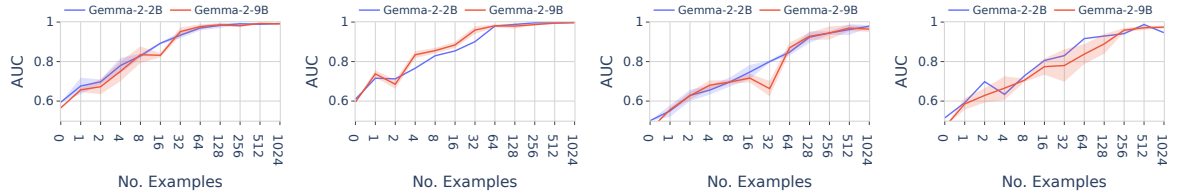
Mistral, Language L_1 | Mistral, Language L_2 | Mistral, Language L_4 | Mistral, Language L_5



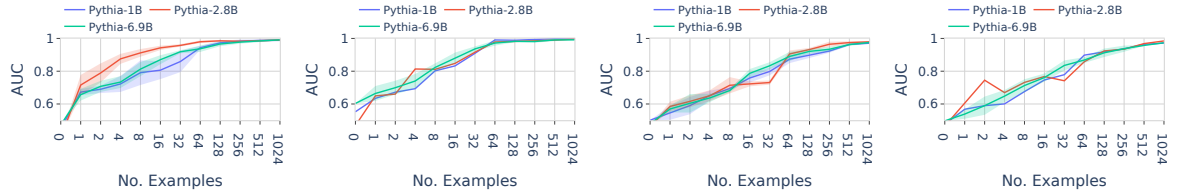
Llama-2, Language L_1 | Llama-2, Language L_2 | Llama-2, Language L_4 | Llama-2, Language L_5



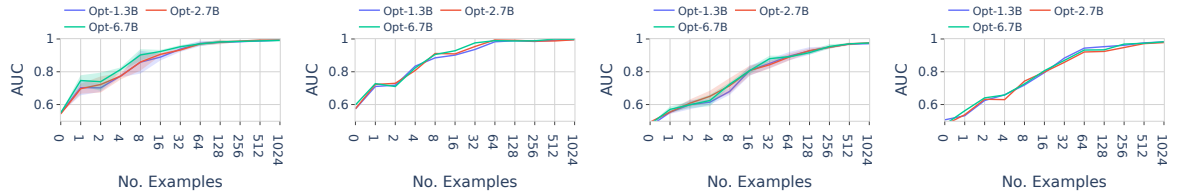
Llama-3, Language L_1 | Llama-3, Language L_2 | Llama-3, Language L_4 | Llama-3, Language L_5



Gemma-, Language L_1 | Gemma, Language L_2 | Gemma, Language L_4 | Gemma, Language L_5

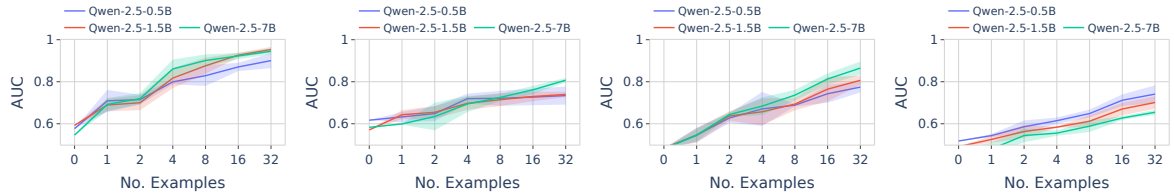


Pythia, Language L_1 | Pythia, Language L_2 | Pythia, Language L_4 | Pythia, Language L_5

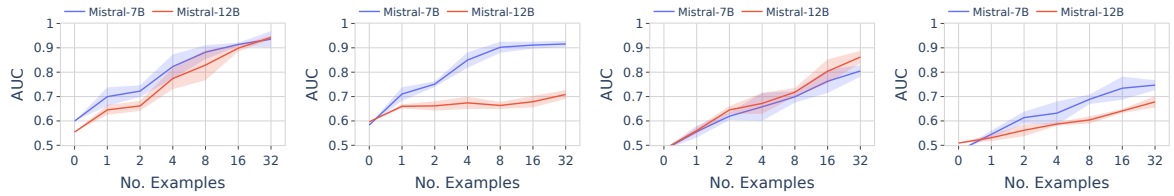


Opt, Language L_1 | Opt, Language L_2 | Opt, Language L_4 | Opt, Language L_5

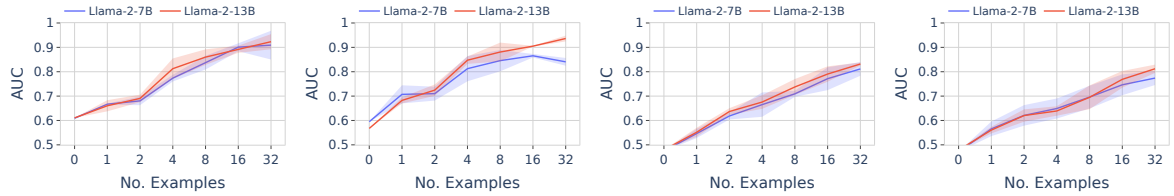
Figure 17: Intra-family FT performance.



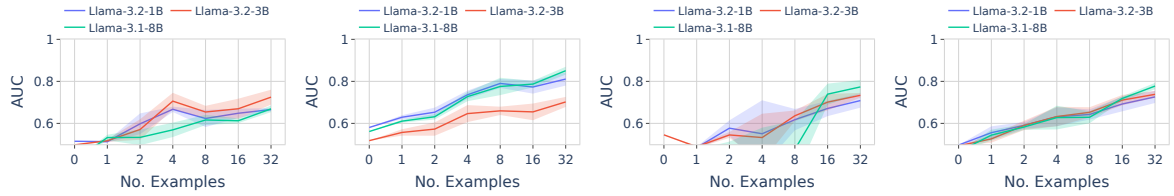
Qwen, Language L_1 | Qwen, Language L_2 | Qwen, Language L_4 | Qwen, Language L_5



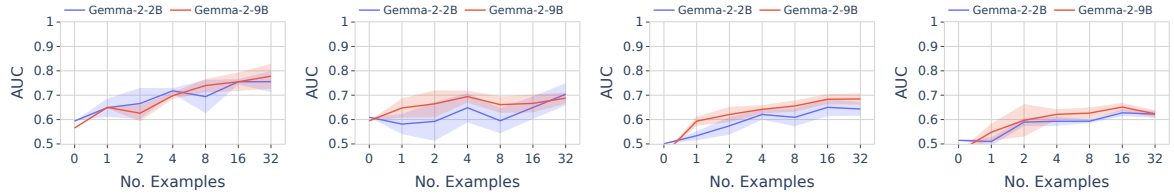
Mistral, Language L_1 | Mistral, Language L_2 | Mistral, Language L_4 | Mistral, Language L_5



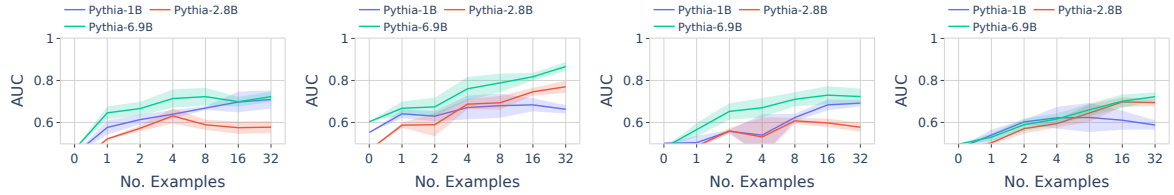
Llama-2, Language L_1 | Llama-2, Language L_2 | Llama-2, Language L_4 | Llama-2, Language L_5



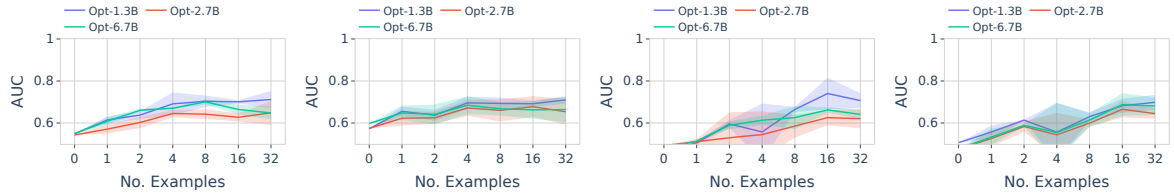
Llama-3, Language L_1 | Llama-3, Language L_2 | Llama-3, Language L_4 | Llama-3, Language L_5



Gemma-, Language L_1 | Gemma, Language L_2 | Gemma, Language L_4 | Gemma, Language L_5

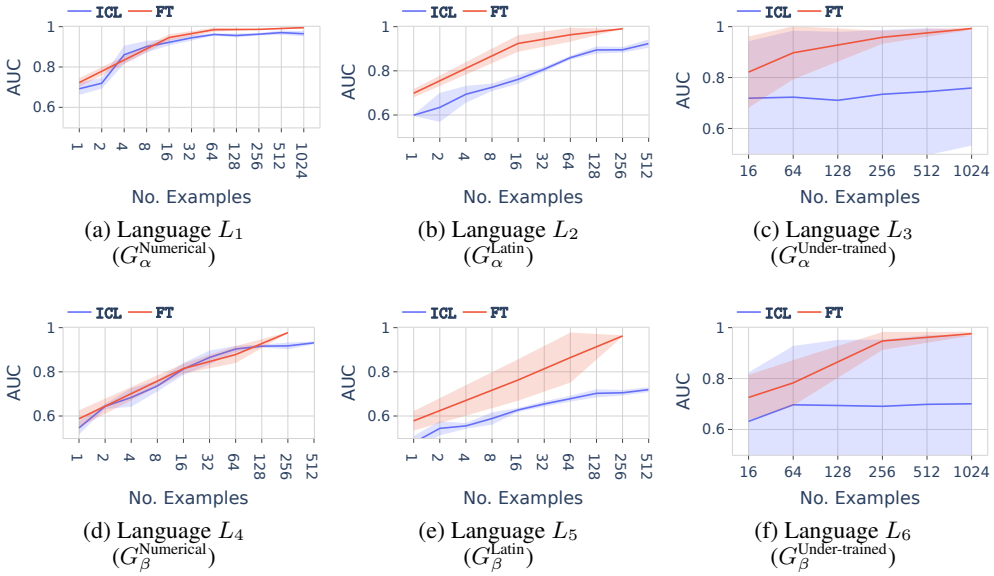


Pythia, Language L_1 | Pythia, Language L_2 | Pythia, Language L_4 | Pythia, Language L_5



Opt, Language L_1 | Opt, Language L_2 | Opt, Language L_4 | Opt, Language L_5

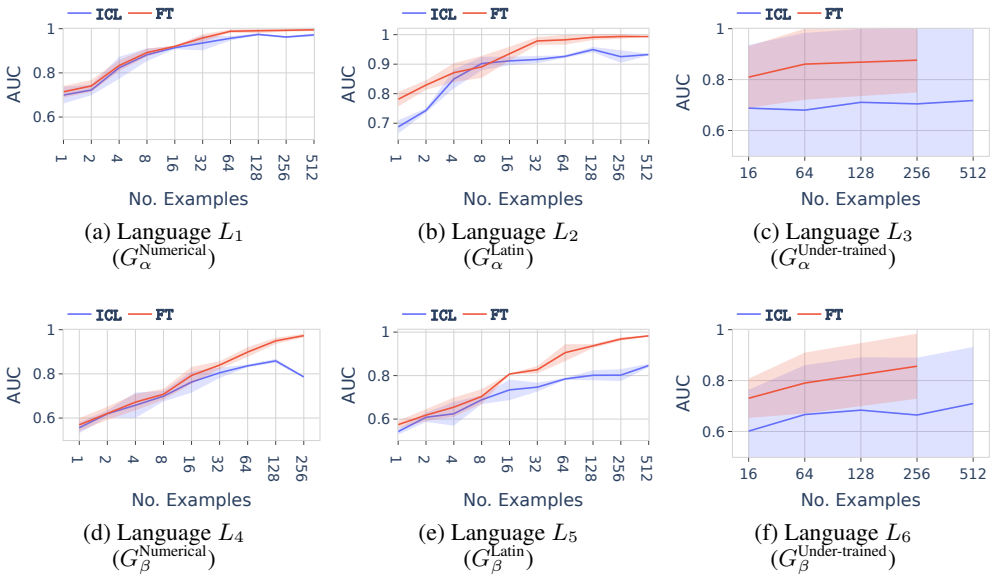
Figure 18: Intra-family ICL performance.



(a) Language L_1 ($G_\alpha^{\text{Numerical}}$) (b) Language L_2 (G_α^{Latin}) (c) Language L_3 ($G_\alpha^{\text{Under-trained}}$)

(d) Language L_4 ($G_\beta^{\text{Numerical}}$) (e) Language L_5 (G_β^{Latin}) (f) Language L_6 ($G_\beta^{\text{Under-trained}}$)

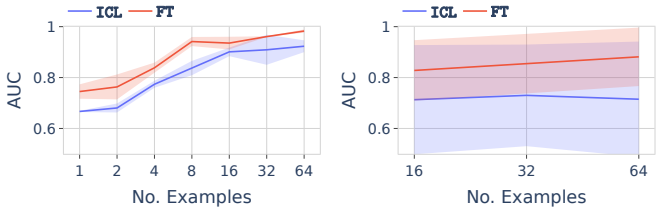
Figure 19: Qwen-2.5-7B: comparison between fine-tuning and in-context learning across different languages



(a) Language L_1 ($G_\alpha^{\text{Numerical}}$) (b) Language L_2 (G_α^{Latin}) (c) Language L_3 ($G_\alpha^{\text{Under-trained}}$)

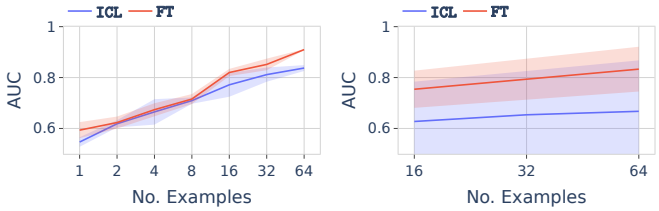
(d) Language L_4 ($G_\beta^{\text{Numerical}}$) (e) Language L_5 (G_β^{Latin}) (f) Language L_6 ($G_\beta^{\text{Under-trained}}$)

Figure 20: Mistral-7B: comparison between fine-tuning and in-context learning across different languages



(a) Language L_1 ($G_\alpha^{\text{Numerical}}$)

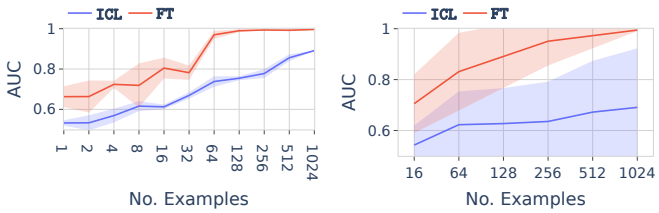
(b) Language L_3 ($G_\alpha^{\text{Under-trained}}$)



(c) Language L_4 ($G_\beta^{\text{Numerical}}$)

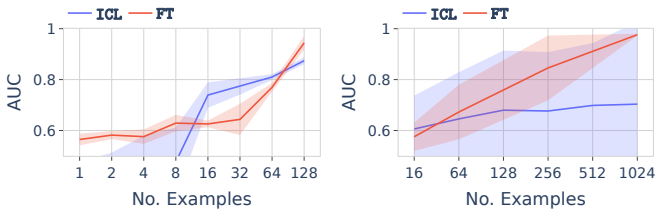
(d) Language L_6 ($G_\beta^{\text{Under-trained}}$)

Figure 21: Llama-2-7B: comparison between fine-tuning and in-context learning across different languages.



(a) Language L_1 ($G_\alpha^{\text{Numerical}}$)

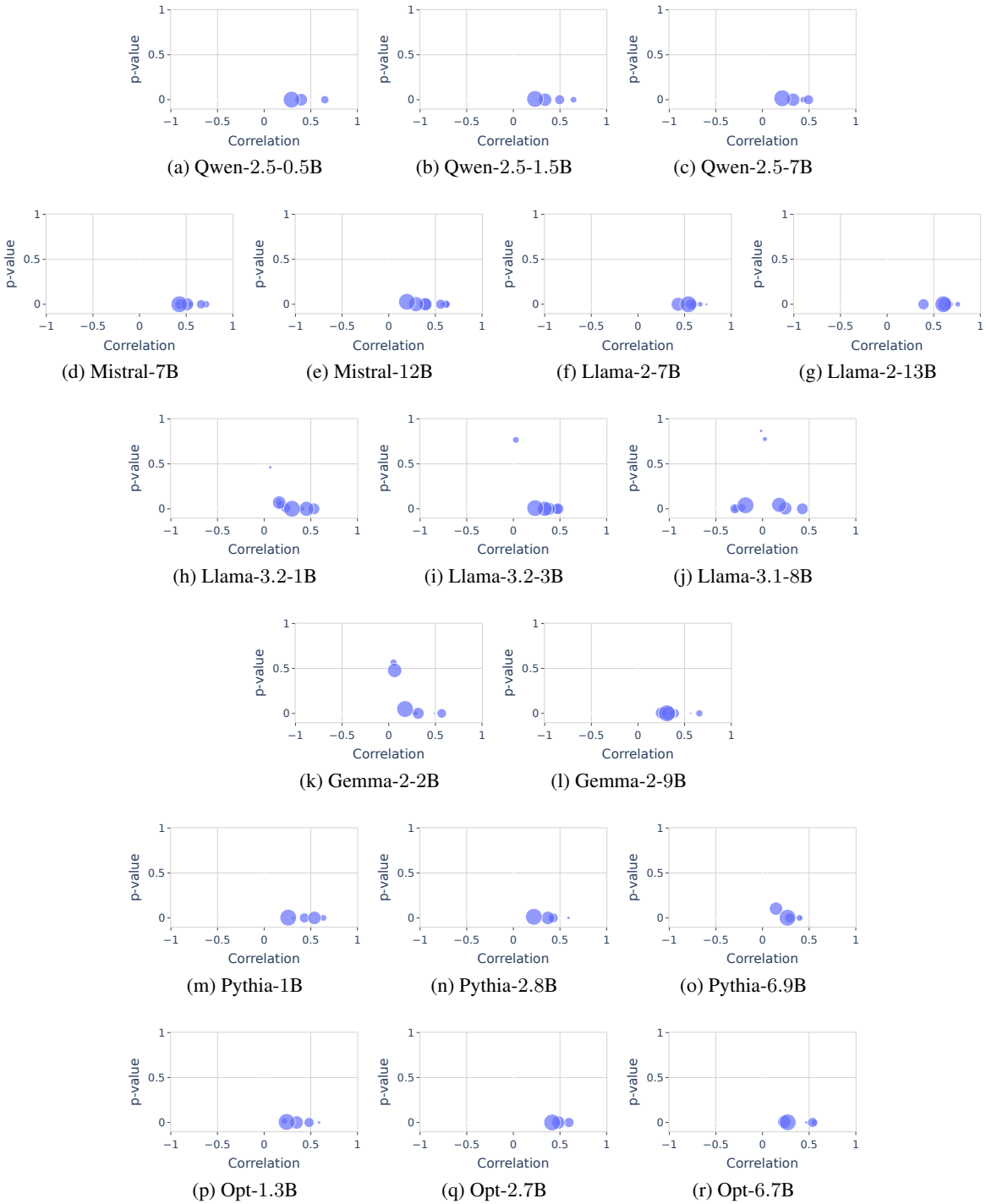
(b) Language L_3 ($G_\alpha^{\text{Under-trained}}$)



(c) Language L_4 ($G_\beta^{\text{Numerical}}$)

(d) Language L_6 ($G_\beta^{\text{Under-trained}}$)

Figure 22: Llama-3.1-8B: comparison between fine-tuning and in-context learning across different languages



(a) Qwen-2.5-0.5B (b) Qwen-2.5-1.5B (c) Qwen-2.5-7B

(d) Mistral-7B (e) Mistral-12B (f) Llama-2-7B (g) Llama-2-13B

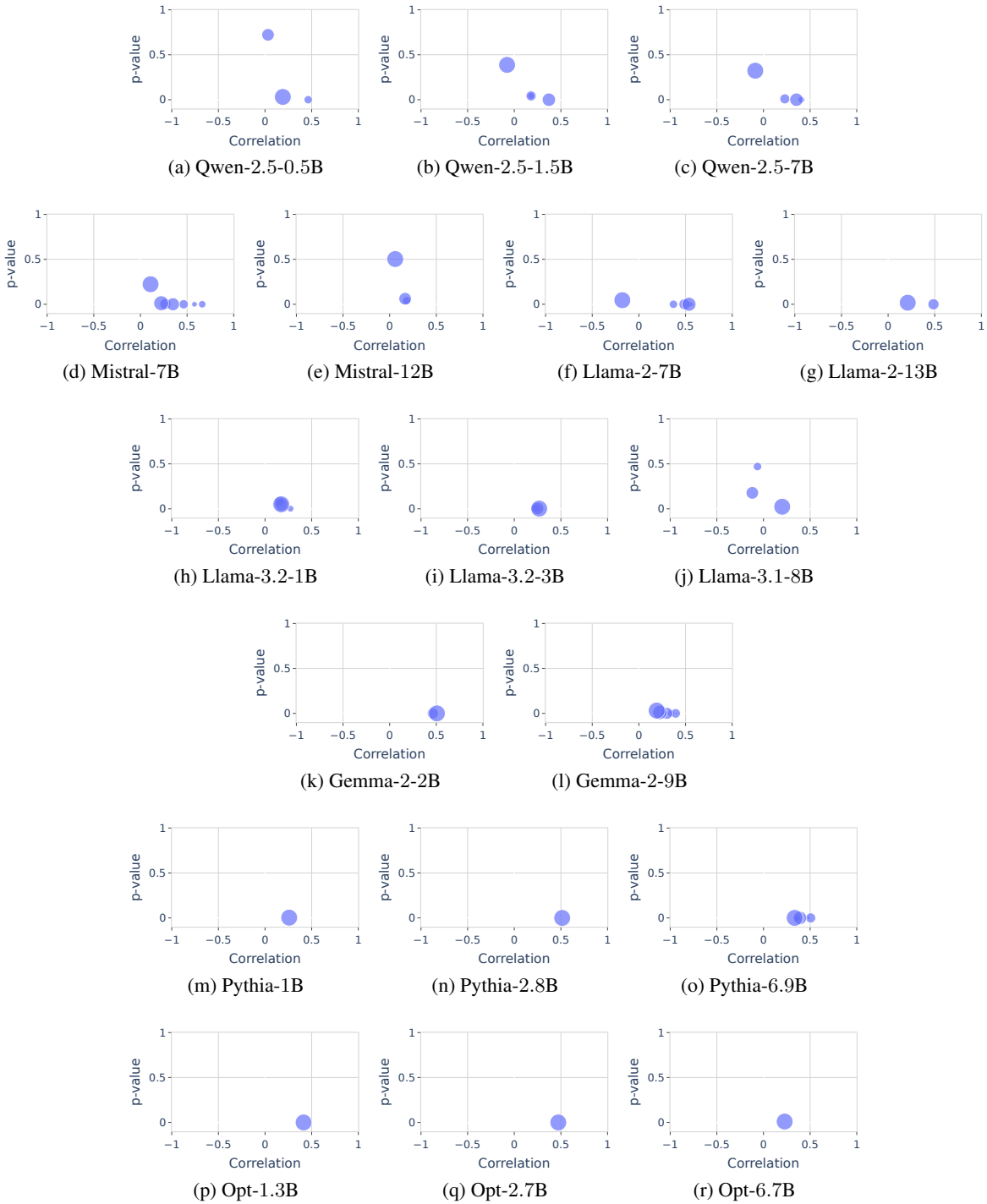
(h) Llama-3.2-1B (i) Llama-3.2-3B (j) Llama-3.1-8B

(k) Gemma-2-2B (l) Gemma-2-9B

(m) Pythia-1B (n) Pythia-2.8B (o) Pythia-6.9B

(p) Opt-1.3B (q) Opt-2.7B (r) Opt-6.7B

Figure 23: Inductive bias of ICL and FT on language L_1, computed as the Pearson correlation of generation loss of FT and ICL on identical test strings. Correlation, despite being positive, tends to decrease with higher examples (larger markers).



(a) Qwen-2.5-0.5B (b) Qwen-2.5-1.5B (c) Qwen-2.5-7B

(d) Mistral-7B (e) Mistral-12B (f) Llama-2-7B (g) Llama-2-13B

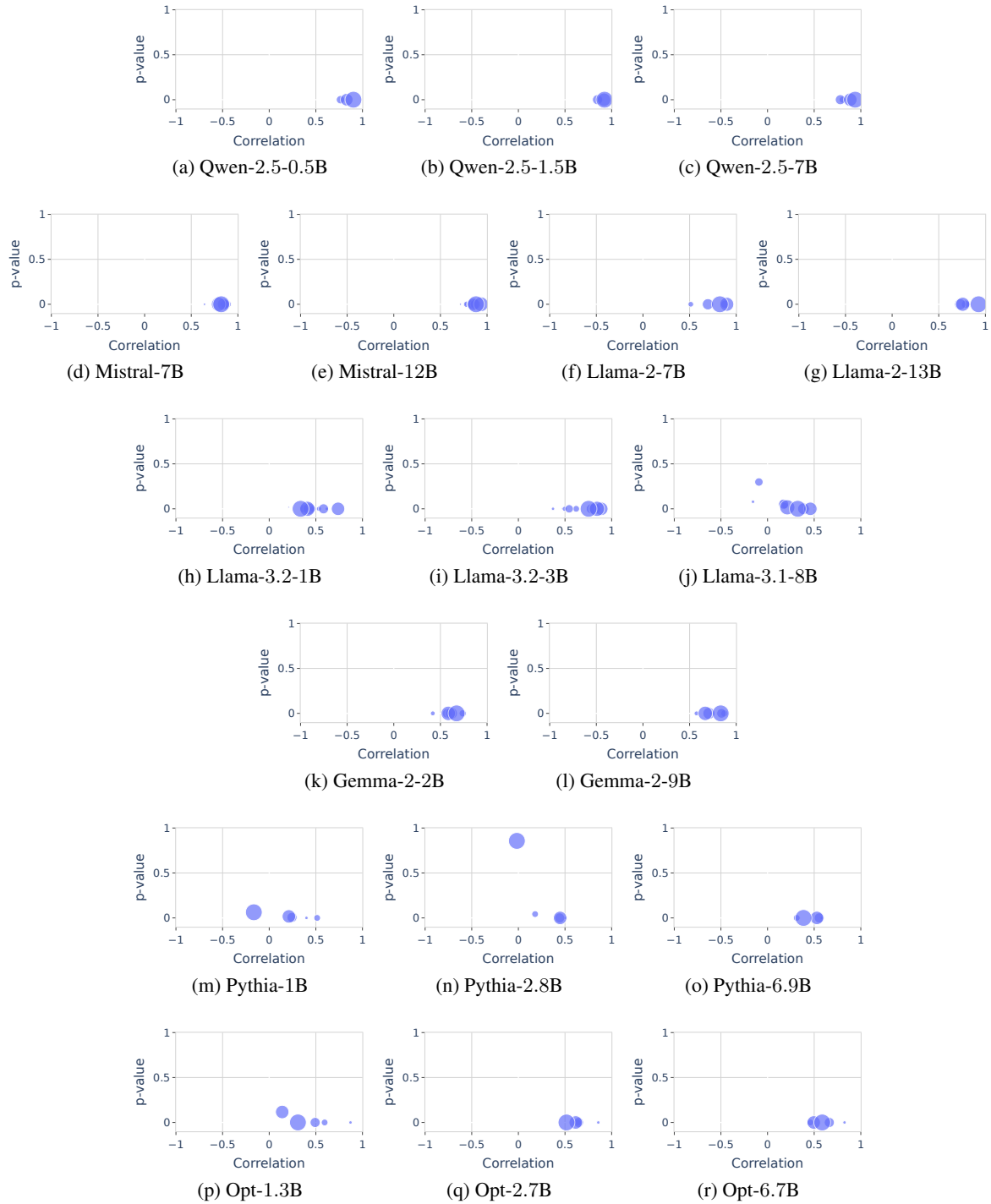
(h) Llama-3.2-1B (i) Llama-3.2-3B (j) Llama-3.1-8B

(k) Gemma-2-2B (l) Gemma-2-9B

(m) Pythia-1B (n) Pythia-2.8B (o) Pythia-6.9B

(p) Opt-1.3B (q) Opt-2.7B (r) Opt-6.7B

Figure 24: Inductive bias of ICL and FT on language L_2, computed as the Pearson correlation of generation loss of FT and ICL on identical test strings. Correlation, despite being positive, tends to decrease with higher examples (larger markers).



(a) Qwen-2.5-0.5B (b) Qwen-2.5-1.5B (c) Qwen-2.5-7B

(d) Mistral-7B (e) Mistral-12B (f) Llama-2-7B (g) Llama-2-13B

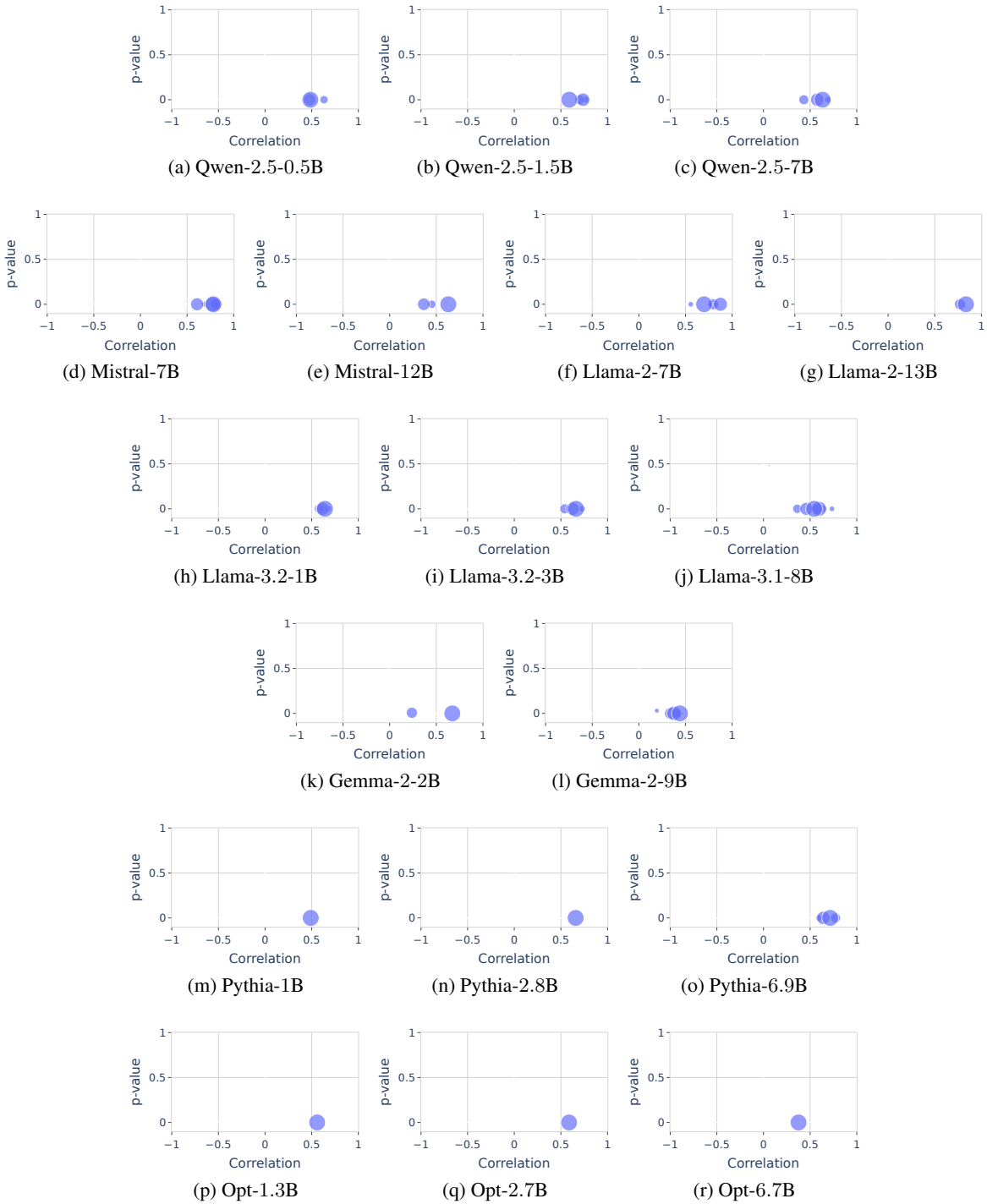
(h) Llama-3.2-1B (i) Llama-3.2-3B (j) Llama-3.1-8B

(k) Gemma-2-2B (l) Gemma-2-9B

(m) Pythia-1B (n) Pythia-2.8B (o) Pythia-6.9B

(p) Opt-1.3B (q) Opt-2.7B (r) Opt-6.7B

Figure 25: Inductive bias of ICL and FT on language L_4, computed as the Pearson correlation of generation loss of FT and ICL on identical test strings. Correlation, despite being positive, tends to decrease with higher examples (larger markers).



(a) Qwen-2.5-0.5B (b) Qwen-2.5-1.5B (c) Qwen-2.5-7B

(d) Mistral-7B (e) Mistral-12B (f) Llama-2-7B (g) Llama-2-13B

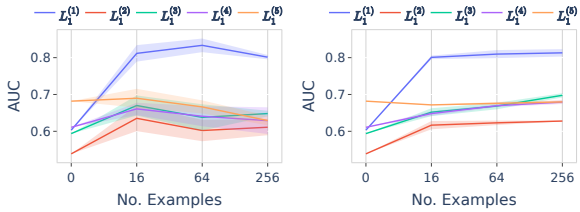
(h) Llama-3.2-1B (i) Llama-3.2-3B (j) Llama-3.1-8B

(k) Gemma-2-2B (l) Gemma-2-9B

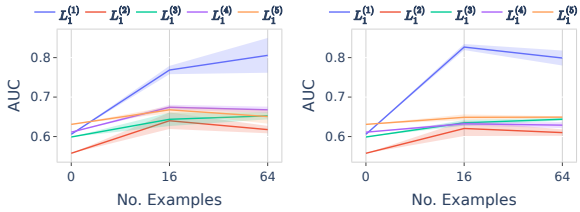
(m) Pythia-1B (n) Pythia-2.8B (o) Pythia-6.9B

(p) Opt-1.3B (q) Opt-2.7B (r) Opt-6.7B

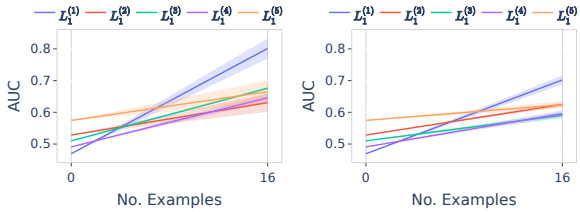
Figure 26: Inductive bias of ICL and FT on language L_5, computed as the Pearson correlation of generation loss of FT and ICL on identical test strings. Correlation, despite being positive, tends to decrease with higher examples (larger markers).



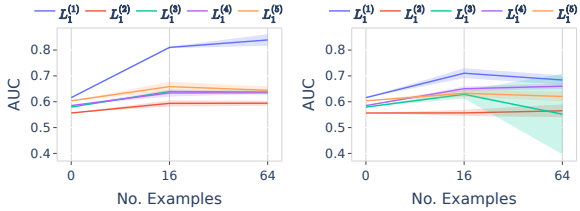
(a) FT, Mistral-7B (b) ICL, Mistral-7B



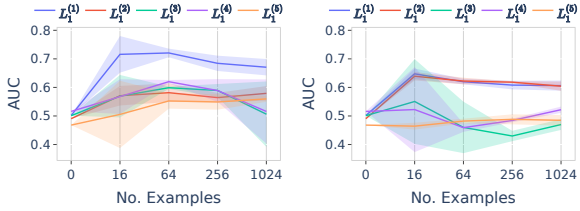
(c) FT, Llama-2-7B (d) ICL, Llama-2-7B



(e) FT, Pythia-6.9B (f) ICL, Pythia-6.9B



(g) FT, Gemma-2-2B (h) ICL, Gemma-2-2B



(i) FT, Llama-3.2-1B (j) ICL, Llama-3.2-1B

Figure 27: Out-of-distribution generalization to languages of increasing distance using FT and ICL. We consider L_1 as the base language. We create languages of higher distance, denoted by $L_1^{(\ell)}$, by changing ℓ production rules in the grammar of L_1. $L_1^{(\ell)}$ contains all changed rules in $L_1^{(\ell-1)}$. Hence, $\texttt{dist}(L_1, L_1^{(1)}) \leq \cdots \leq \texttt{dist}(L_1, L_1^{(5)})$ (see Eq. (1))

D Fine-tuning vs. In-context Learning on Natural Language Datasets

We conduct a comparison of FT and ICL on a natural language dataset to observe whether our findings on formal languages generalize to natural language datasets. We consider a natural language inference task on the MNLI dataset (Williams et al., 2018), as studied in the related work by Mosbach et al. (2023). The learning objective is to generate the sentiment label given the premise and hypothesis (see Table 3).

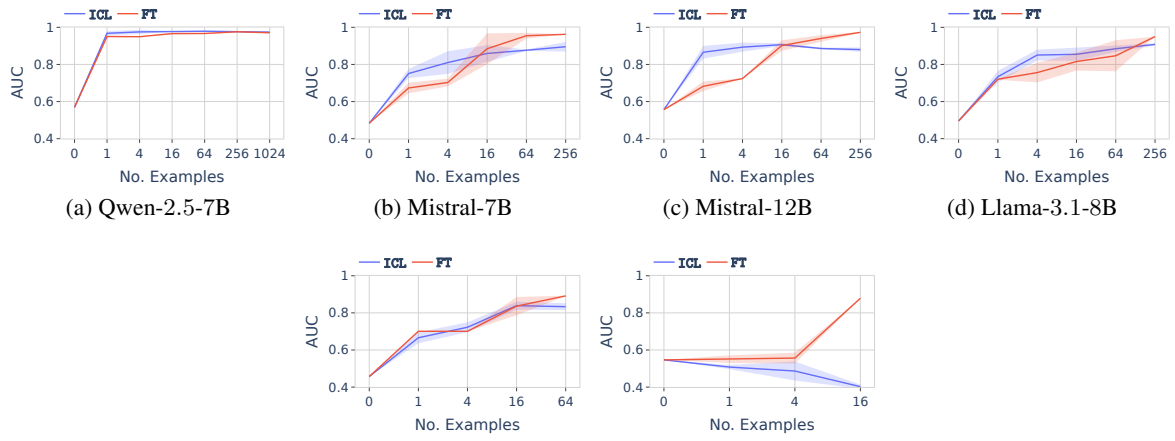
Issues of Data Contamination. In Figure 28, FT surpasses ICL on the MNLI dataset with increasing examples, *which is consistent with our findings on formal languages* in Figure 7. However, Qwen-2.5-7B model performs much better than other models in both learning modes, suggesting the *possibility* of data contamination. As evidence, the MNLI dataset was proposed in 2018, which is earlier than the release of Qwen-2.5-7B model in 2024. Therefore, it is difficult to fairly compare different models or their learning modes on publicly available datasets, if a subset of models is possibly trained on the testing dataset (Dominguez-Olmedo et al., 2025). This further strengthens our case that *synthetic formal languages should be adopted widely to critically evaluate the performance of LLMs, where the risk of data contamination is minimal*.

Difficulty in Identifying In-distribution vs. Out-of-distribution Tasks. In Figure 29, we demonstrate in-distribution and out-of-distribution performance side-by-side on the MNLI dataset for both learning modes. The differentiation of tasks is determined by the genre of (premise, hypothesis) pairs. If the genre of the testing pair matches with training pairs, then the task is in-distribution. Otherwise, the task is out-of-distribution. However, we do not observe any difference in the comparison of FT vs. ICL based on tasks – FT is better than ICL in both tasks. This contradicts our findings in formal languages in Figure 8, where the distance between tasks is well-defined, and both FT and ICL perform equally well in the out-of-distribution task. This experiment highlights the ambiguity of specifying learning tasks in natural language datasets, the core theme in desideratum **D1** in Section 1. *Therefore, for an objective comparison, it is important to carefully define in-distribution and out-of-distribution tasks, which is easier in formal languages than natural languages.*

Table 3: Construction of in-language and out-language strings for the MNLI dataset (Williams et al., 2018), where the out-language string differs from the in-language string only in the sentiment label. The discriminative test is successful, if the generation loss of the correct label in the in-language string is lower than that of the incorrect label in the out-language string. The prompt instruction is shown in the table below.

In-language string	Out-language string (edit at label)
Premise: One of our number will carry out your instructions minutely. **Hypothesis**: A member of my team will execute your orders with immense precision. **Label**: entailment	**Premise**: One of our number will carry out your instructions minutely. **Hypothesis**: A member of my team will execute your orders with immense precision. **Label**: neutral
Premise: Fun for adults and children. **Hypothesis**: Fun for only children. **Label**: contradiction	**Premise**: Fun for adults and children. **Hypothesis**: Fun for only children. **Label**: entailment

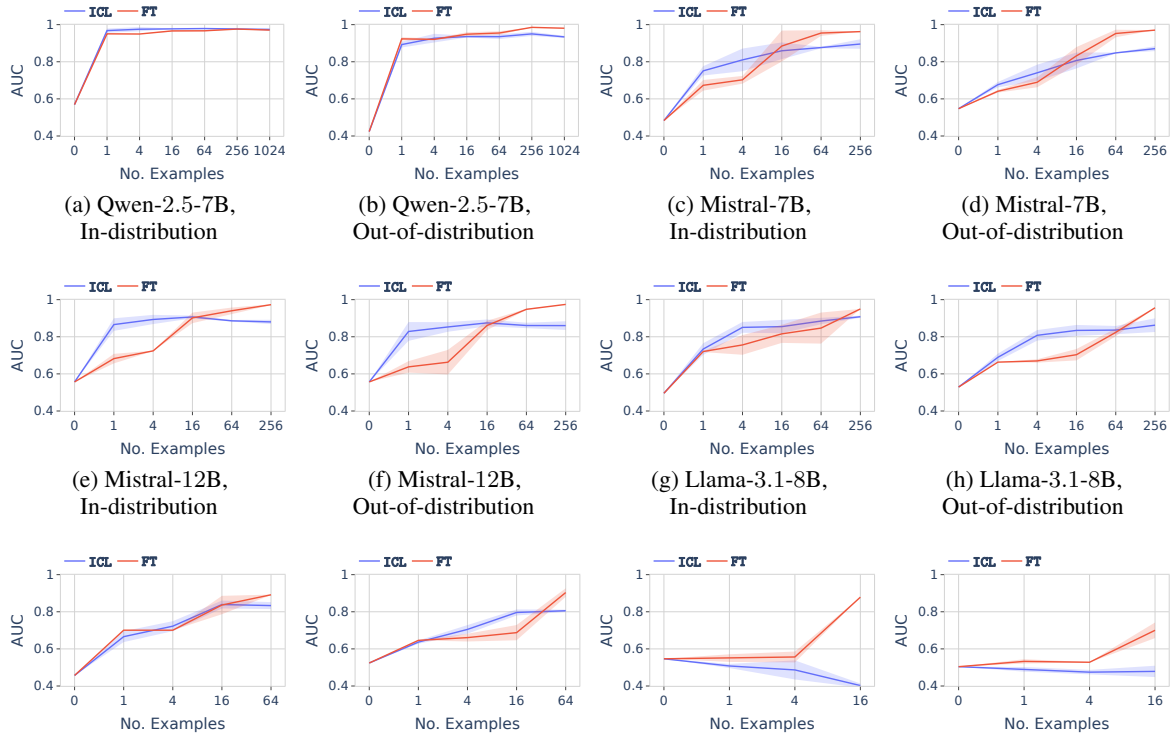
Prompt Instruction (beginning of the prompt)
Provide a classification label for the pair, indicating the relationship between the premise and hypothesis: - entailment : The hypothesis logically follows from the premise. - neutral : The hypothesis is neither entailed nor contradicted by the premise. - contradiction : The hypothesis contradicts the premise.



(a) Qwen-2.5-7B (b) Mistral-7B (c) Mistral-12B (d) Llama-3.1-8B

(e) Llama-2-13B (f) Opt-6.7B

Figure 28: In-distribution generalization of FT and ICL on the MNLI dataset, where the learning task is to perform natural language inference by generating the sentiment label {entailment, neutral, contradiction} given premise and hypothesis. At a high level, FT is better than ICL with more examples, consistent with results on formal languages. In a detailed analysis, we observe that different LLMs perform differently given the same problem, indicating the possibility of data contamination in some well-performing LLMs, such as Qwen-2.5-7B.



(a) Qwen-2.5-7B, In-distribution
(b) Qwen-2.5-7B, Out-of-distribution
(c) Mistral-7B, In-distribution
(d) Mistral-7B, Out-of-distribution
(e) Mistral-12B, In-distribution
(f) Mistral-12B, Out-of-distribution
(g) Llama-3.1-8B, In-distribution
(h) Llama-3.1-8B, Out-of-distribution
(i) Llama-2-13B, In-distribution
(j) Llama-2-13B, Out-of-distribution
(k) Opt-6.7B, In-distribution
(l) Opt-6.7B, Out-of-distribution

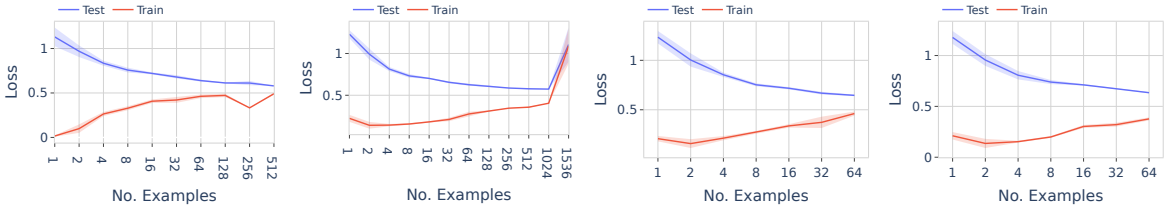
Figure 29: MNLI dataset: In-distribution (inference within the same genre, Column 1 and 3) vs. out-of-distribution (inference across genres, Column 2 and 4) generalization performance of FT and ICL, where there is no substantial difference across tasks. This is a fundamental problem in natural language datasets, where the identification of tasks can be ambiguous, and LLMs may not distinguish them. Overall, FT is better than ICL, which contradicts our results on formal languages where FT is better than ICL in in-distribution generalization only, but both learning modes perform equally well in out-of-distribution generalization.

E Testing the Limit of In-context Learning

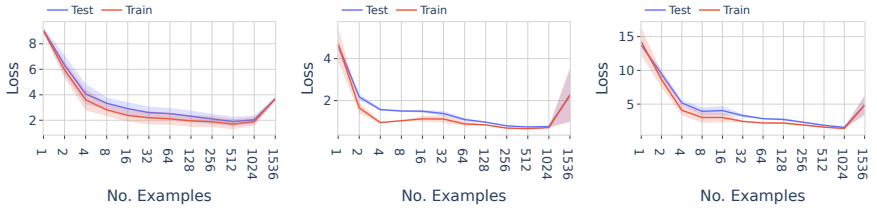
To find the limit of ICL ability of an LLM, we rely on the convergence of training and test loss in ICL as examples are added. Intuitively, training loss provides a practical *lower bound* on test loss in ICL. An LLM can no longer improve in ICL when both losses converge. To obtain training loss, we first provide ICL examples from the training set and later compute the loss of generating each training example already present in the context.

Empirically, across all languages, test loss converges to train loss, i.e., ICL *limit is reached* in the majority of LLMs, except in the Llama-2 and Opt families. These two families have limited context (4K and 2K tokens, respectively), and there is a gap between losses even upon exhausting their context length. Moreover, long-context LLMs, such as Qwen-2.5-7B and Llama-3.1-8B with 128K context length, cannot further improve from additional examples as both losses converge and later increase near the limit (see Figure 30). *Therefore, formal language learning enables us to categorize LLMs into two: (a) LLMs that cannot reach the* ICL *limit, and (b) LLMs that reach their* ICL *limit and do not improve with additional examples towards their context length limit.*

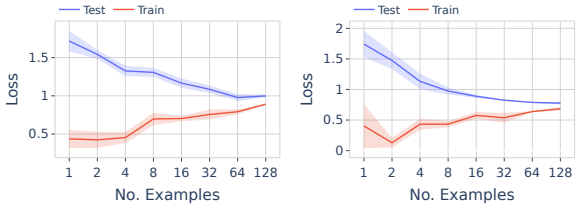
(a) Qwen-2.5-0.5B (b) Qwen-2.5-1.5B (c) Qwen-2.5-7B



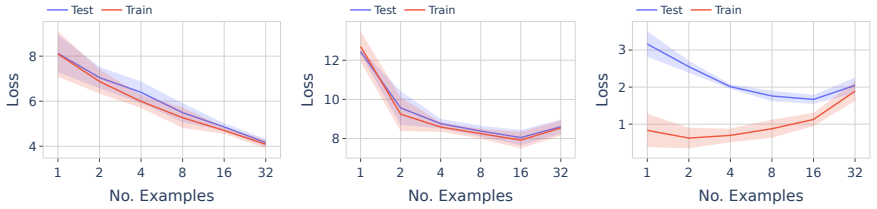
(d) Mistral-7B (e) Mistral-12B (f) Llama-2-7B (g) Llama-2-13B



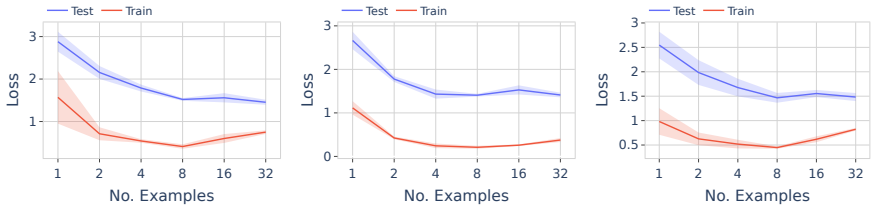
(h) Llama-3.2-1B (i) Llama-3.2-3B (j) Llama-3.1-8B



(k) Gemma-2-2B (l) Gemma-2-9B

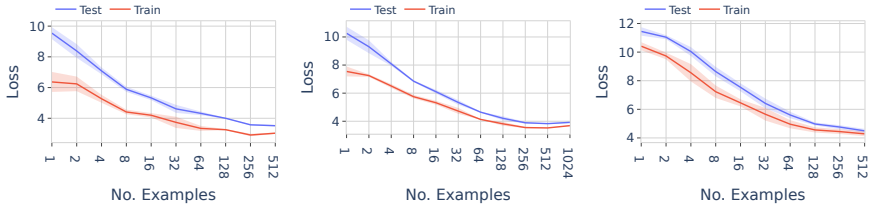


(m) Pythia-1B (n) Pythia-2.8B (o) Pythia-6.9B

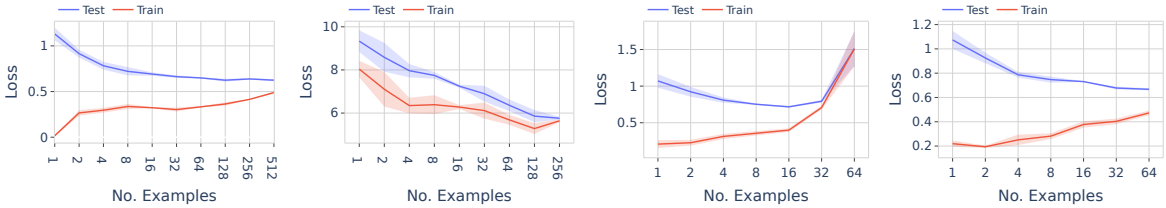


(p) Opt-1.3B (q) Opt-2.7B (r) Opt-6.7B

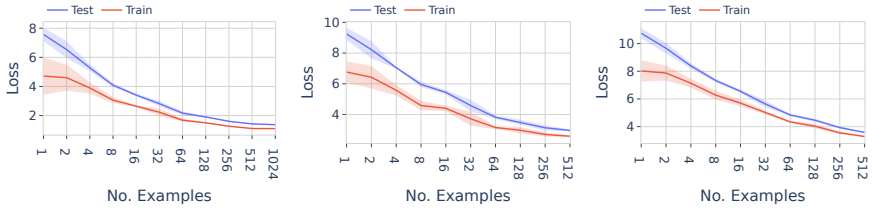
Figure 30: Testing the limit of utilizing ICL context (1536 examples $\approx$ 77K tokens) on language L_1. Training loss provides a lower bound of test loss in ICL. Long context LLMs cannot further improve from additional examples.



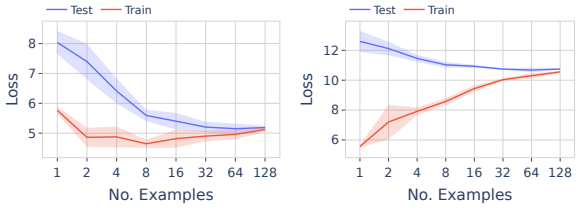
(a) Qwen-2.5-0.5B (b) Qwen-2.5-1.5B (c) Qwen-2.5-7B



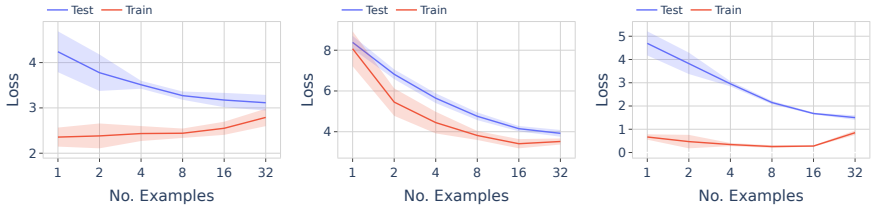
(d) Mistral-7B (e) Mistral-12B (f) Llama-2-7B (g) Llama-2-13B



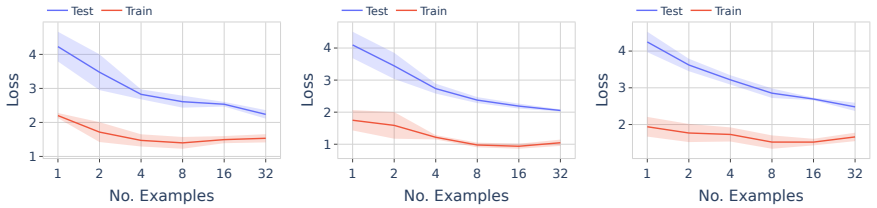
(h) Llama-3.2-1B (i) Llama-3.2-3B (j) Llama-3.1-8B



(k) Gemma-2-2B (l) Gemma-2-9B

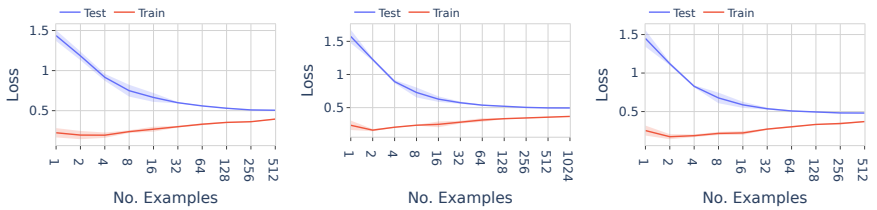


(m) Pythia-1B (n) Pythia-2.8B (o) Pythia-6.9B

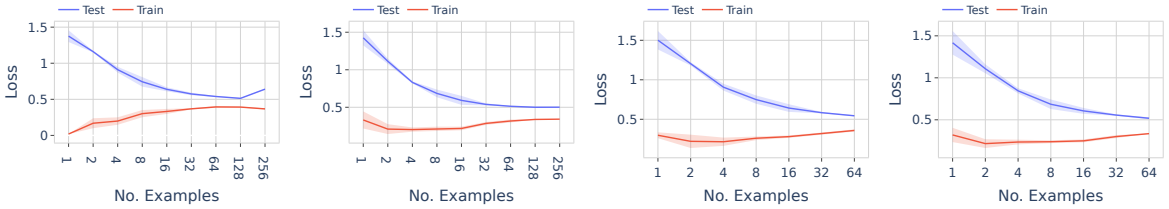


(p) Opt-1.3B (q) Opt-2.7B (r) Opt-6.7B

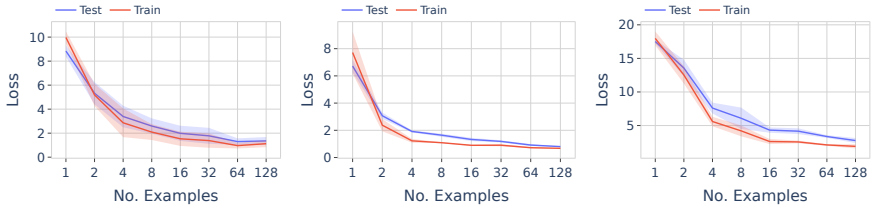
Figure 31: Testing the limit of utilizing ICL context (1536 examples $\approx$ 77K tokens) on language L_2. Training loss provides a lower bound of test loss in ICL. Long context LLMs cannot further improve from additional examples.



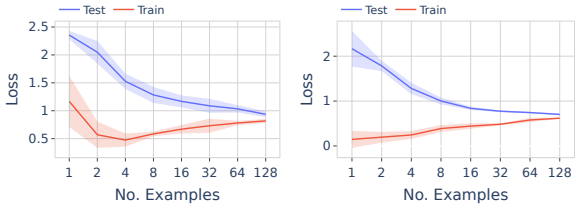
(a) Qwen-2.5-0.5B (b) Qwen-2.5-1.5B (c) Qwen-2.5-7B



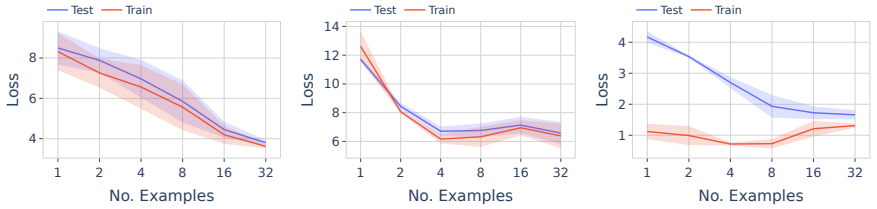
(d) Mistral-7B (e) Mistral-12B (f) Llama-2-7B (g) Llama-2-13B



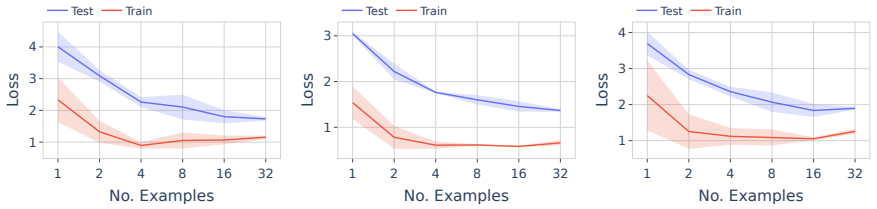
(h) Llama-3.2-1B (i) Llama-3.2-3B (j) Llama-3.1-8B



(k) Gemma-2-2B (l) Gemma-2-9B

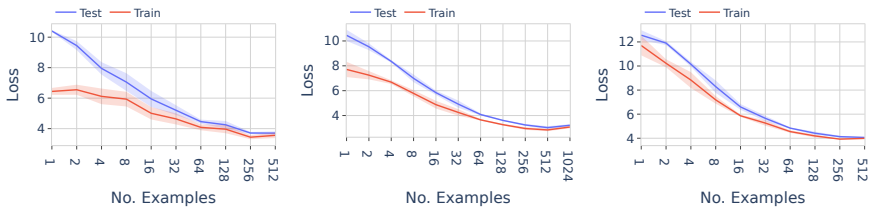


(m) Pythia-1B (n) Pythia-2.8B (o) Pythia-6.9B

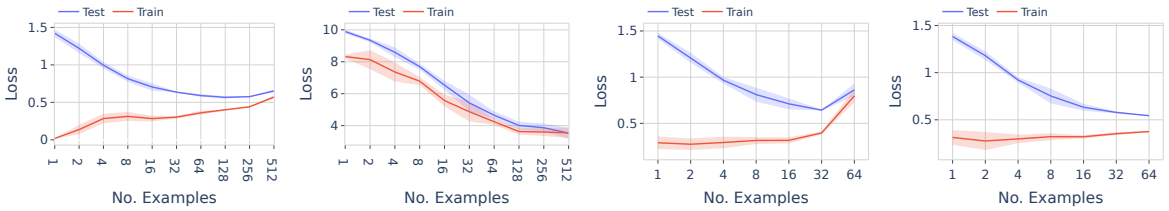


(p) Opt-1.3B (q) Opt-2.7B (r) Opt-6.7B

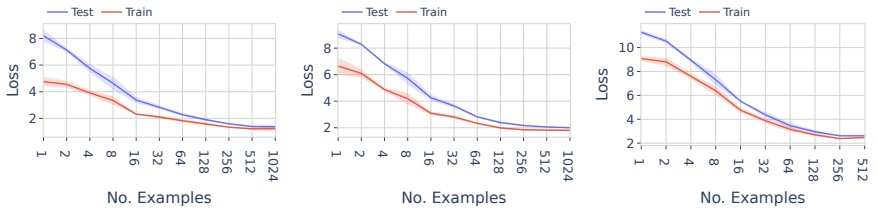
Figure 32: Testing the limit of utilizing ICL context on language L_4. Training loss provides a lower bound of test loss in ICL. Long context LLMs cannot further improve from additional examples.



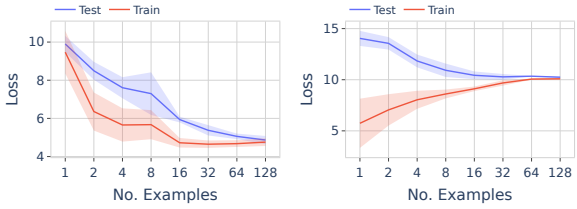
(a) Qwen-2.5-0.5B (b) Qwen-2.5-1.5B (c) Qwen-2.5-7B



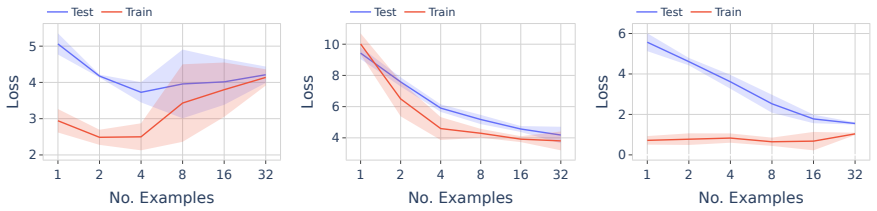
(d) Mistral-7B (e) Mistral-12B (f) Llama-2-7B (g) Llama-2-13B



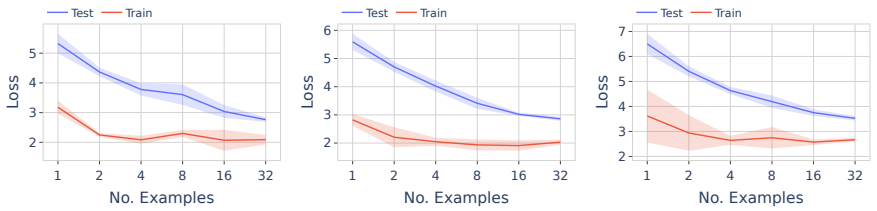
(h) Llama-3.2-1B (i) Llama-3.2-3B (j) Llama-3.1-8B



(k) Gemma-2-2B (l) Gemma-2-9B



(m) Pythia-1B (n) Pythia-2.8B (o) Pythia-6.9B



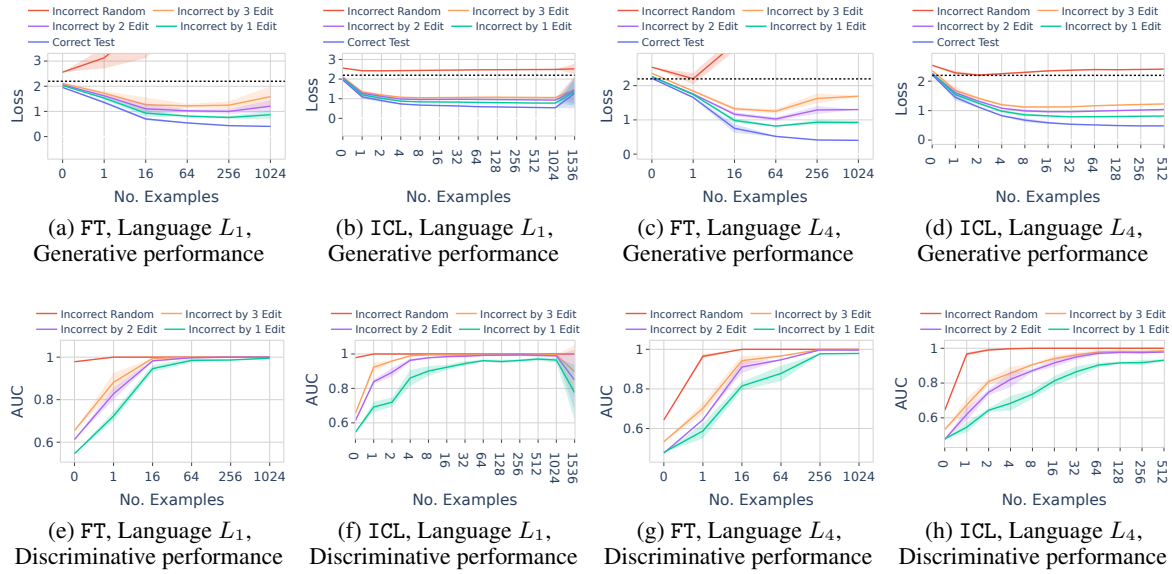
(p) Opt-1.3B (q) Opt-2.7B (r) Opt-6.7B

Figure 33: Testing the limit of utilizing ICL context on language L_5. Training loss provides a lower bound of test loss in ICL. Long context LLMs cannot further improve from additional examples.

F Discriminative Test

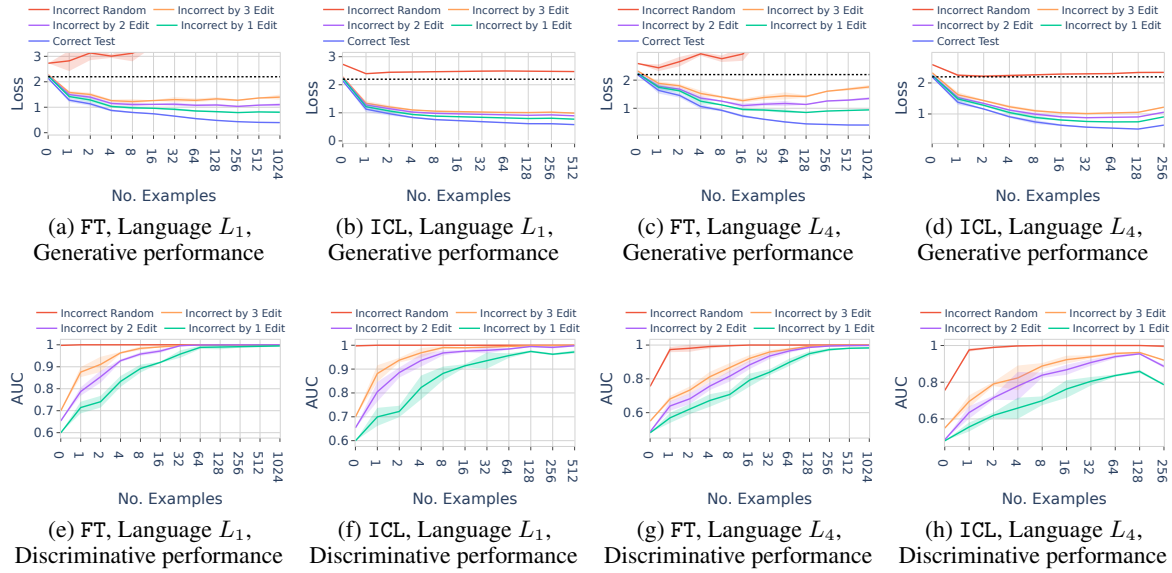
Claim 1**.** *For a given language, the discriminative test yields a numerically comparable score between two learning modes of an LLM and across LLMs, unlike the generative test.*

The discriminative test computes a classification score to determine how well strings in a language are discriminated from strings outside the language, using generation loss. Crucially, within each learning mode, both in-language and out-language strings undergo the same input formatting and are evaluated under the same parameters and hyperparameters of the LLM. For example, in `ICL`, the same concatenated prefix is applied to all strings; in `FT`, all strings use a null prefix (see Figure 1). As a result, any mode-specific or model-specific bias in generation loss is cancelled out in the relative comparison, making the classification score numerically comparable across learning modes and LLMs. The generative test, however, compares absolute generation loss on in-language strings only. Since LLMs with different pretraining priors assign different baseline generation loss to the same strings, generative scores are not numerically comparable across LLMs.



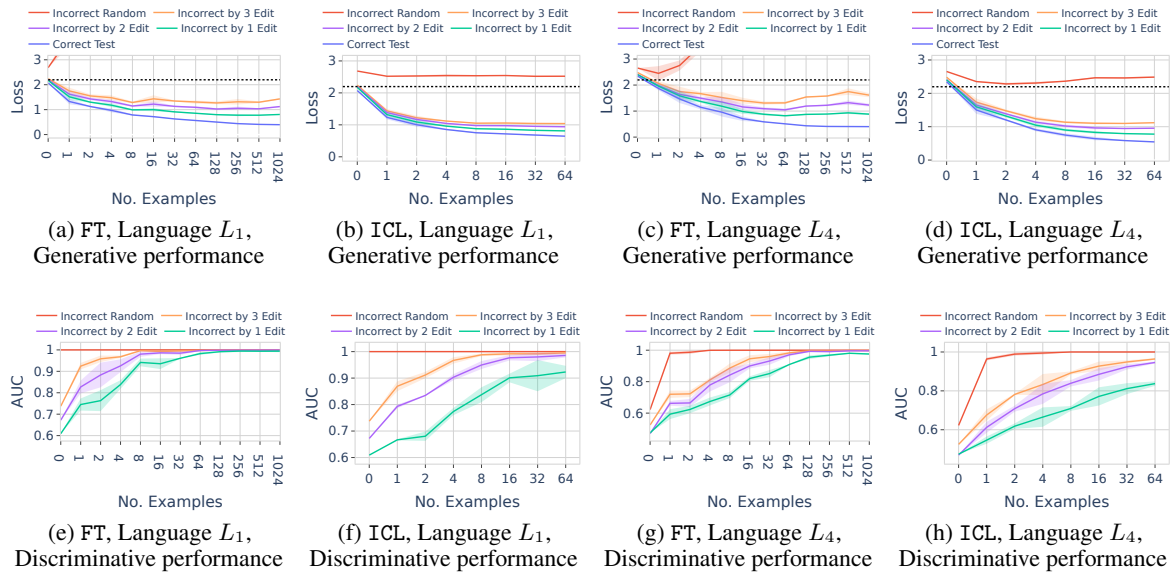
(a) FT, Language L_1, Generative performance
(b) ICL, Language L_1, Generative performance
(c) FT, Language L_4, Generative performance
(d) ICL, Language L_4, Generative performance
(e) FT, Language L_1, Discriminative performance
(f) ICL, Language L_1, Discriminative performance
(g) FT, Language L_4, Discriminative performance
(h) ICL, Language L_4, Discriminative performance

Figure 34: Qwen-2.5-7B: Language proficiency according to generative (first row) and discriminative (second row) tests. First two columns are for language L_1, and the last two columns are for language L_4.



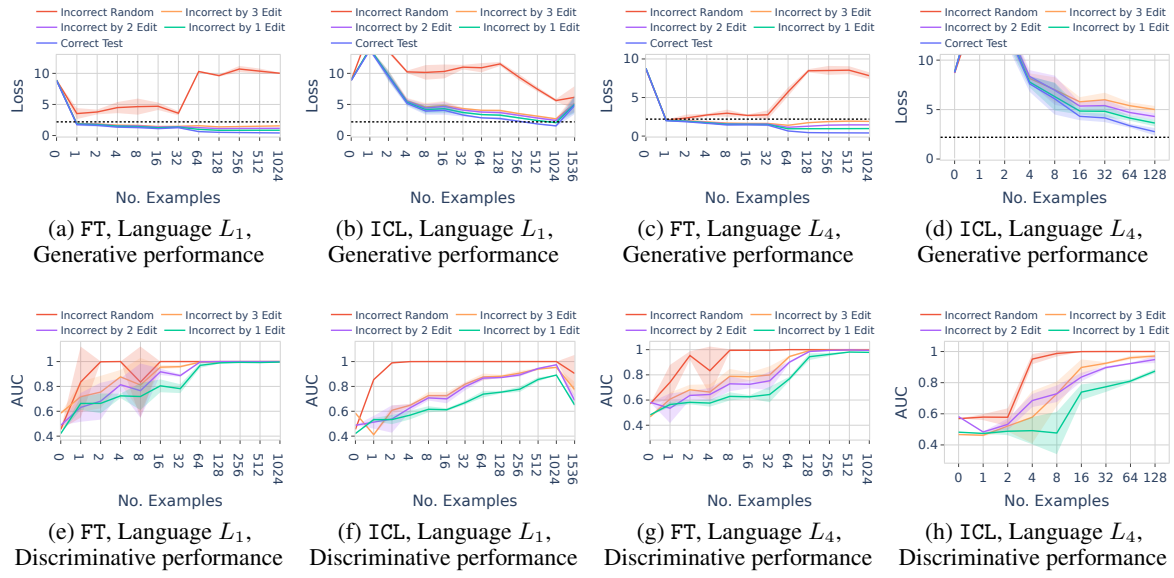
(a) FT, Language L_1, Generative performance
(b) ICL, Language L_1, Generative performance
(c) FT, Language L_4, Generative performance
(d) ICL, Language L_4, Generative performance
(e) FT, Language L_1, Discriminative performance
(f) ICL, Language L_1, Discriminative performance
(g) FT, Language L_4, Discriminative performance
(h) ICL, Language L_4, Discriminative performance

Figure 35: Mistral-7B: Language proficiency according to generative (first row) and discriminative (second row) tests. First two columns are for language L_1, and the last two columns are for language L_4.



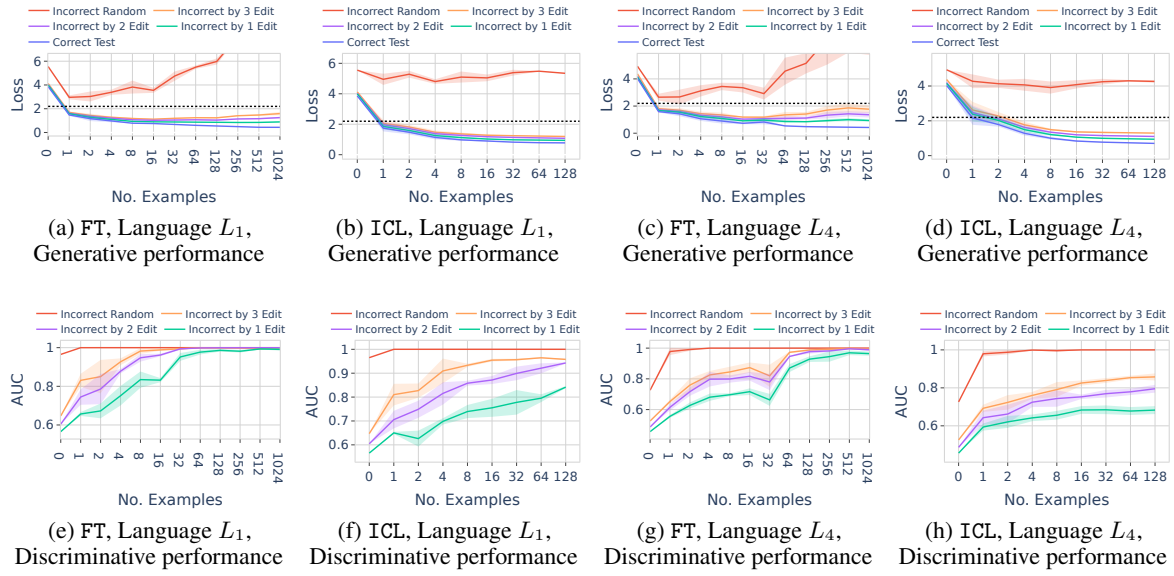
(a) FT, Language L_1, Generative performance
(b) ICL, Language L_1, Generative performance
(c) FT, Language L_4, Generative performance
(d) ICL, Language L_4, Generative performance
(e) FT, Language L_1, Discriminative performance
(f) ICL, Language L_1, Discriminative performance
(g) FT, Language L_4, Discriminative performance
(h) ICL, Language L_4, Discriminative performance

Figure 36: Llama-2-7B: Language proficiency according to generative (first row) and discriminative (second row) tests. First two columns are for language L_1, and the last two columns are for language L_4.



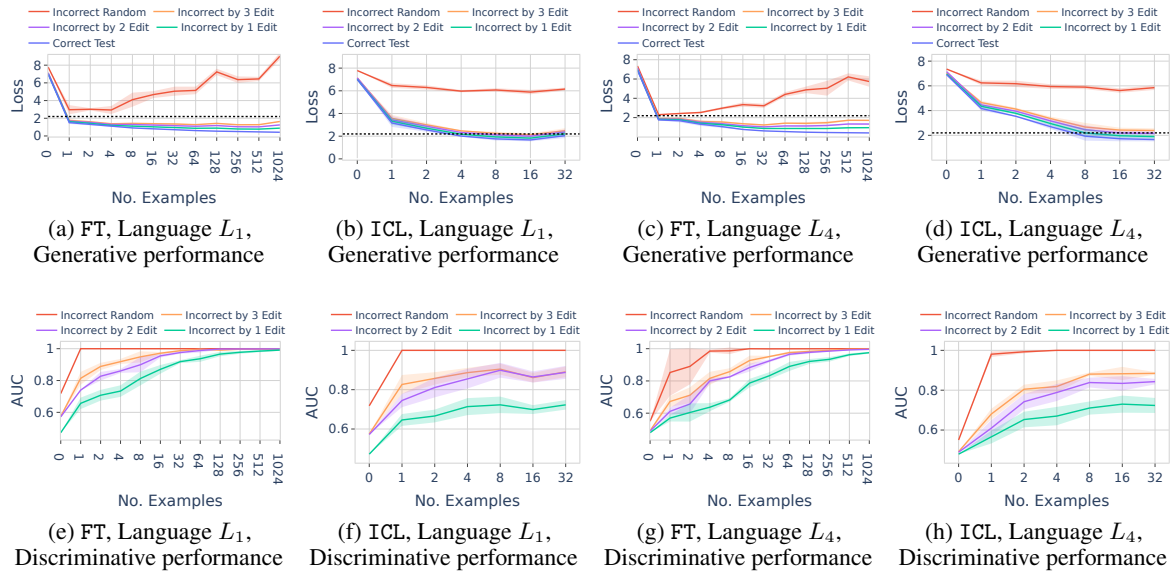
(a) FT, Language L_1, Generative performance
(b) ICL, Language L_1, Generative performance
(c) FT, Language L_4, Generative performance
(d) ICL, Language L_4, Generative performance
(e) FT, Language L_1, Discriminative performance
(f) ICL, Language L_1, Discriminative performance
(g) FT, Language L_4, Discriminative performance
(h) ICL, Language L_4, Discriminative performance

Figure 37: Llama-3.1-8B: Language proficiency according to generative (first row) and discriminative (second row) tests. First two columns are for language L_1, and the last two columns are for language L_4.



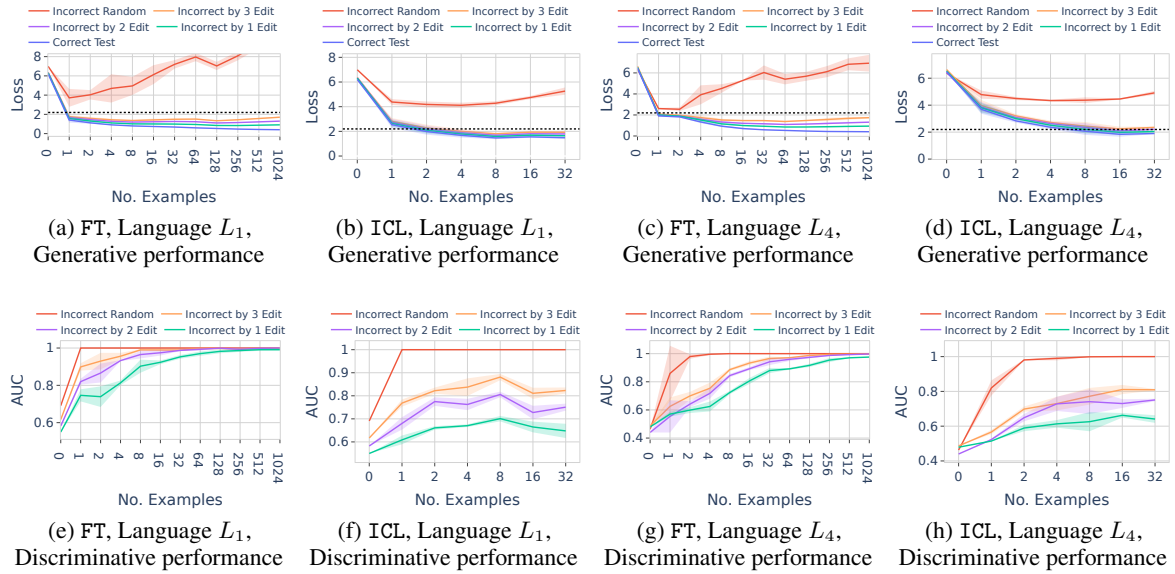
(a) FT, Language L_1, Generative performance
(b) ICL, Language L_1, Generative performance
(c) FT, Language L_4, Generative performance
(d) ICL, Language L_4, Generative performance
(e) FT, Language L_1, Discriminative performance
(f) ICL, Language L_1, Discriminative performance
(g) FT, Language L_4, Discriminative performance
(h) ICL, Language L_4, Discriminative performance

Figure 38: Gemma-2-9B: Language proficiency according to generative (first row) and discriminative (second row) tests. First two columns are for language L_1, and the last two columns are for language L_4.



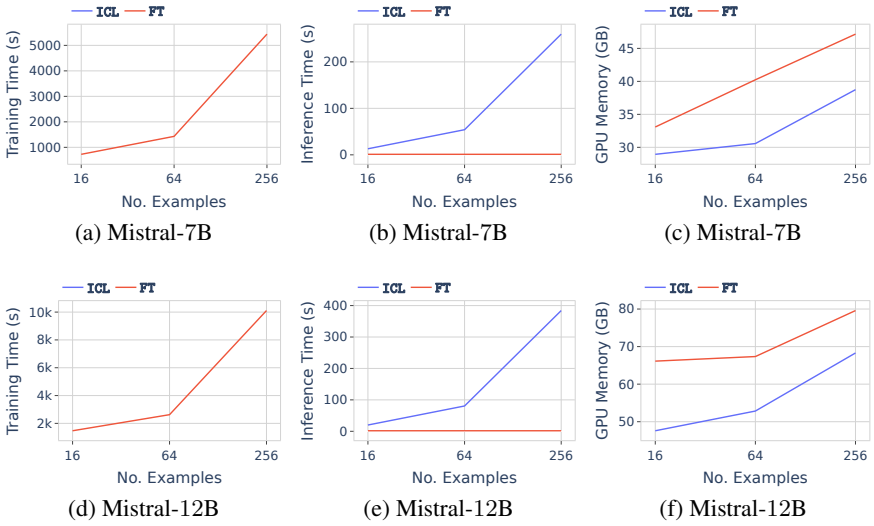
(a) FT, Language L_1, Generative performance
(b) ICL, Language L_1, Generative performance
(c) FT, Language L_4, Generative performance
(d) ICL, Language L_4, Generative performance
(e) FT, Language L_1, Discriminative performance
(f) ICL, Language L_1, Discriminative performance
(g) FT, Language L_4, Discriminative performance
(h) ICL, Language L_4, Discriminative performance

Figure 39: Pythia-6.9B: Language proficiency according to generative (first row) and discriminative (second row) tests. First two columns are for language L_1, and the last two columns are for language L_4.



(a) FT, Language L_1, Generative performance
(b) ICL, Language L_1, Generative performance
(c) FT, Language L_4, Generative performance
(d) ICL, Language L_4, Generative performance
(e) FT, Language L_1, Discriminative performance
(f) ICL, Language L_1, Discriminative performance
(g) FT, Language L_4, Discriminative performance
(h) ICL, Language L_4, Discriminative performance

Figure 40: Opt-6.7B: Language proficiency according to generative (first row) and discriminative (second row) tests. First two columns are for language L_1, and the last two columns are for language L_4.



(a) Mistral-7B
(b) Mistral-7B
(c) Mistral-7B
(d) Mistral-12B
(e) Mistral-12B
(f) Mistral-12B

Figure 41: Comparing FT and ICL across compute cost, such as training cost (column 1), inference cost (column 2), and memory cost (column 3), recorded for language L_1. The results show that FT and ICL are expensive in different phases of computation: FT incurs training cost, which does not apply to ICL. In contrast, ICL has significantly higher inference cost, despite requiring less memory. Our paper therefore compares both learning modes on equal data budget, using the same number of examples for training and inference.

G Implications of the Study

We elaborate on the implications of our findings across four research questions in Section 5. We provide our hypothesis for each finding, which may inspire future research.

- In **RQ1**, when learning a language, FT performance converges across LLMs but ICL performance is variable. Our hypothesis is that FT is a direct form of learning, where parameters are explicitly updated toward the task. Since we evaluate FT at its optimal performance across epochs, and the considered language is simple with a hierarchical recursive structure, all LLMs reach a similar performance ceiling, leading to convergence across LLMs.

  ICL, however, is an indirect form of learning, where the model infers patterns from the context without any parameter update. As a result, ICL performance retains model-specific pre-training biases, which differ across LLMs of different sizes and families, leading to variable performance. A more subtle analysis is given below in **RQ4**.

- In **RQ2**, FT outperforms ICL in in-distribution generalization, where the training and test languages are the same. In formal languages, however, both modes perform equally in out-of-distribution generalization, generalizing only to languages closer to the training language. Therefore, if the test language is different, FT is no longer the better mode, and explicit parameter update in FT does not help. In this case, ICL is a more natural choice, since its parameters remain unchanged and no specialization toward the training language occurs, unlike FT where parameter updates specialize the model toward the training language, without improving out-of-distribution generalization.

- In **RQ3**, the inductive bias of FT and ICL is similar, but this similarity often decreases with more training examples, where similarity is measured by the Pearson correlation of generation losses on identical test strings. Thus, when learning is stronger, each mode learns the language differently, leading to different inductive biases.

- In **RQ4**, ICL is less robust than FT across languages. Since ICL relies on pre-training, its performance depends on how well the pre-training corpus covers the specific tokens of the language, making it sensitive to token variation. FT avoids this problem by directly updating parameters toward the language, leading to more robust performance.

- We emphasize the adoption of the discriminative test for evaluating language proficiency in LLMs, across both formal and natural languages. The discriminative test ensures that in-language strings are generated with higher probability than, and are even separable from, out-of-language strings – a stronger condition than the generative test on in-language strings only.

  For future work on the adoption of the discriminative test, one needs to systematically generate strings outside the language, which we have shown for formal languages in Section 3, and for natural language, with sentiment classification as one instance, in Appendix D. Since natural language is less well-defined than formal language, the boundary between in-language and out-of-language strings may be harder to define precisely in natural language, warranting careful study.

H Use of AI Assistants

We use AI assistants for the following purposes:

- Paper writing: We use Claude to correct grammatical mistakes, and paraphrase sentences to improve the quality and flow of the writing.

- Coding: We use Claude and Windsurf as coding assistants.

Nevertheless, we take full responsibility for the content of the paper and code.